SAS® Inventory Replenishment Planning 2.3 User's Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2012. *SAS® Inventory Replenishment Planning 2.3 User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Inventory Replenishment Planning 2.3 User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

June 2018

Contents

About SAS Inventory Replenishment Planning Documentation

Credits

Documentation

Writing	Xinmin Wu, Michelle Opp, Daniel Underwood, Tugrul Sanli, Jinxin Yi
Editing	Anne Baxter, Ed Huddleston
Documentation Support	Tim Arnold

Software

SAS Inventory Replenishment Planning is implemented by the following staff of Operations Research Department in the Advanced Analytics Division. Other members of the division have given substantial support to the project.

IRP procedure	Xinmin Wu, Tugrul Sanli
MIRP procedure	Xinmin Wu, Michelle Opp, Daniel Underwood, Jinxin Yi

Support Groups

Software Testing	Matthew Fischl, Daniel Underwood, Lan Li
Technical Support	Tonya Chapman

Acknowledgments

Many people have been instrumental in the development of SAS Inventory Replenishment Planning. We would like to thank Professor Paul Zipkin at Duke University for his invaluable comments.

The final responsibility for the SAS System lies with SAS alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

Chapter 1

What's New in SAS Inventory Replenishment Planning 2.3

In SAS Inventory Replenishment Planning 2.3, the MIRP procedure provides more features and greater functionality than it did in the previous release. There is no change in the IRP procedure.

New Functionality in the MIRP Procedure

The following functionality was added to the MIRP procedure in SAS Inventory Replenishment Planning 2.3:

- The new SYSTEM=PUSH option enables optimal inventory distribution during large-scale promotional or seasonal sales. Going beyond merely satisfying customer demand, this new option finds an optimized way to push excess inventory to different retail locations. Retailer inventory-capacity constraints can be enforced when the SYSTEM=PUSH option is used by the new variable named Capacity in the node data set.
- The new node data set variable named MPL (for *minimum presentation level*) enables inventory optimization to keep at least a specified minimum on-hand inventory at retail locations.
- Inventory amounts can now be negative at period 1 in order to represent initial backlog.

Chapter 2
The IRP Procedure

Contents

Overview

The IRP procedure provides the ability to calculate periodic-review inventory replenishment policies for single-location and two-echelon-distribution inventory systems. These policies are determined through a number of algorithms that are controlled by user-specified options.

PROC IRP can calculate four types of replenishment policies that are different types of (s, S) or (s, NQ) policies. For more information, see the section "Replenishment Policies" on page 27.

An *optimal* policy is defined as a policy that minimizes the average cost—the total of ordering, holding, and back-order penalty costs. PROC IRP uses several heuristic algorithms to approximate optimal policies to meet the user-specified service constraints. If the penalty cost information is available, PROC IRP can also calculate optimal policies for single-location inventory systems.

Getting Started

Single-Location Inventory Systems

In a single-location inventory system, customers (or demand transactions) request a random amount of an item (SKU). Customer orders are filled from on-hand inventory. If insufficient inventory is available, the order is filled partially with available inventory and any unsatisfied portion is backlogged (or back-ordered). The *inventory position*, which is on-hand inventory plus on-order inventory minus back orders, is monitored periodically. Based on the current inventory position, the replenishment policy determines whether or not a replenishment order should be placed from an outside supplier.

Periodic review is the most common type of review process. Inventory is counted or evaluated periodically (for example, monthly) at discrete points in time to determine whether a replenishment order needs to be placed. Replenishment decisions can be made only at those points. The time between two review points is called the *review period*.

The delay between when a replenishment order is placed and when the order arrives is called the *lead time* and is specified in the same units as the review period. For example, if the review period is one day (that is, inventory is reviewed daily) and the lead time is one week, the lead time is specified as seven days. The IRP procedure accounts for demand that occurs during the lead time.

The size of the demand that occurs during one review period is called the *review-time demand*. When demand is stationary (that is, demand stays relatively constant across review periods), PROC IRP requires only the mean and variance of review-time demand. For example, these values might be estimates that are calculated by using a forecast engine prior to invoking PROC IRP. When demand is not stationary, information must be provided to PROC IRP about the lead-time demand rather than the review-time demand; for more information about lead-time demand, see the section "LEADTIMEDEMAND Statement" on page 17.

PROC IRP calculates inventory replenishment policies by using this information—inventory position, lead time, and review-time demand—together with user-specified inventory-related costs and policy restrictions.

As a simple example, consider a single store that carries five different items (SKUs), which are ordered from an outside supplier. Calculation of demand forecasts and inventory review is done weekly. The manager wants to calculate (s, S) policies that minimize the expected holding and ordering costs and achieve a target fill rate of 95%. Table 2.1 summarizes the demand, lead time, and cost information for these items. The lead times are expressed in terms of weeks (one, two, or three weeks), because the review period is one week.

Table 2.1 Data Summary

SKU	Holding Cost	Ordering Cost	Lead Time	Mean of Demand	Variance of Demand
A	0.35	90	1	125.1	2170.8
B	0.05	50	2	140.3	1667.7
C	0.12	50	3	116.0	3213.4
D	0.10	75	1	291.8	5212.4
E	0.45	75	2	134.5	1980.5

This information is stored in a data set called skuInfo, which is displayed in Figure 2.1. The mean and variance of the one-period demand are given by the RTDmean and RTDvar variables. The lead time is fixed (that is, it has zero variance) and is specified by the LTmean variable. Similarly, holding and ordering costs are specified by the holdingCost and fixedCost variables. Finally, the serviceLevel variable specifies the desired service level.

Figure 2.1 Input Data Set skuInfo

```
                          Input Data Set

                holding    fixed                                  service
 Obs     sku      Cost      Cost    LTmean    RTDmean    RTDvar     Level

  1       A       0.35       90        1       125.1     2170.8      0.95
  2       B       0.05       50        2       140.3     1667.7      0.95
  3       C       0.12       50        3       116.0     3213.4      0.95
  4       D       0.10       75        1       291.8     5212.4      0.95
  5       E       0.45       75        2       134.5     1980.5      0.95
```

The following IRP procedure call calculates the inventory policies:

```
proc irp data=skuInfo out=policy;
   itemid sku;
   holdingcost holdingCost;
   leadtime / mean=LTmean;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel;
run;
```

The REVIEWTIMEDEMAND statement specifies the variables that contain the mean and variance of review-time demand. Similarly, the LEADTIME statement identifies the variable that contains the lead time, and the SERVICE statement identifies the variable that specifies the desired service levels. Because fill rate is the default service measure and (s, S) policies are the default policy type, no extra options or statements are needed. The variables RTDmean, RTDvar, LTmean, fixedCost, holdingCost, and serviceLevel are all default variable names, so you do not need to specify them in any statements. Thus, the following IRP procedure call would produce the same results as the previous one:

```
proc irp data=skuInfo out=policy;
   itemid sku;
run;
```

The output data set policy is displayed in Figure 2.2.

Figure 2.2 Policy Output Data Set

Output Data Set

Obs	sku	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost	inventory Ratio
1	A	211	463	133.739	6.0735	0.43646	86.0905	1.06906
2	B	335	842	229.279	6.8941	0.29107	26.0175	1.63420
3	C	470	792	216.028	6.0123	0.34361	43.1037	1.86231
4	D	432	1074	282.873	14.4130	0.42098	59.8611	0.96941
5	E	382	597	131.757	6.6193	0.50730	97.3379	0.97961

Obs	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	0.048550	0.93540	0.95155	0.87037	FR-SS-NO	SUCCESSFUL
2	0.049138	0.61192	0.95139	0.88256	FR-SS-NO	SUCCESSFUL
3	0.051830	0.53697	0.95122	0.90925	FR-SS-NO	SUCCESSFUL
4	0.049393	1.03156	0.95062	0.85239	FR-SS-NO	SUCCESSFUL
5	0.049214	1.02082	0.95109	0.86866	FR-SS-NO	SUCCESSFUL

The reorderLevel variable gives the reorder level, s, and the orderUpToLevel variable gives the order-up-to level, S. For example, any time the inventory position for SKU A is observed to be less than or equal to 211 at a review point, a replenishment order is placed to bring the inventory position up to 463. The _STATUS_ variable indicates that the optimization was successful for all observations. The _ALGORITHM_ variable gives information about the algorithm used; namely, a fill rate (FR) service-level heuristic was used to calculate (s, S) policies (SS), using a normal distribution (NO) for lead-time demand and demand during lead time plus review time. The remaining variables report several estimated inventory metrics to evaluate the performance of the underlying policy. For information about these variables, see the section "OUT= Data Set" on page 24.

Two-Echelon-Distribution Inventory Systems

A two-echelon-distribution inventory system consists of a single warehouse and multiple retailer locations. The retailer locations do not incur a fixed cost when they order from the warehouse; therefore, the retailer locations follow a base-stock policy. However, the warehouse incurs a fixed cost when it orders from an outside supplier; the warehouse can therefore follow an (s, S) or (s, nQ) policy. PROC IRP can find nearly optimal policies for two-echelon-distribution inventory systems that have different service constraints on the retailer locations.

Consider a warehouse-retailer distribution problem that has two items. For SKU A, the warehouse is in Raleigh, NC, and the retailer locations are located in Atlanta, GA, Baltimore, MD, and Charleston, SC. For SKU B, the warehouse is in Greensboro, NC, and the retailer locations are in Atlanta, GA, and Charleston, SC. The demand, lead time, and cost information of each item are stored in a data set called skuInfo2, shown in Figure 2.3.

Figure 2.3 Input Data Set skuInfo2

Input Data Set

Obs	sku	warehouse	location	holding Cost
1	A	Raleigh, NC		0.35
2	A	Raleigh, NC	Atlanta, GA	0.70
3	A	Raleigh, NC	Baltimore, MD	0.70
4	A	Raleigh, NC	Charleston, SC	0.70
5	B	Greensboro, NC		0.05
6	B	Greensboro, NC	Atlanta, GA	0.10
7	B	Greensboro, NC	Charleston, SC	0.10

Obs	fixed Cost	LTmean	RTDmean	RTDvar	service Level
1	90	1	125.1	2170.8	.
2	.	2	32.6	460.2	0.95
3	.	2	61.8	1133.5	0.95
4	.	1	30.7	577.1	0.95
5	50	2	140.3	1667.7	.
6	.	2	68.4	907.3	0.95
7	.	1	71.9	760.4	0.95

The location and serviceLevel variables have missing values when the observation corresponds to a warehouse. PROC IRP treats the current observation as a warehouse if the corresponding entry for the location variable is missing. Similarly, the fixedCost variable has missing values for the retailer locations because the retailer locations follow base-stock policies and do not incur ordering costs. Only the warehouses incur ordering costs, because they replenish from an outside supplier.

The following IRP procedure call calculates inventory policies for the warehouses and the retailer locations:

```
proc irp data=skuInfo2 out=policy2;
   itemid sku warehouse;
   location location;
   holdingcost holdingCost;
   leadtime / mean=LTmean;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel;
run;
```

The output data set policy2 is displayed in Figure 2.4. The reorderLevel variable gives the reorder level, s, and the orderUpToLevel variable gives the order-up-to level, S. For the retailers, the order-up-to level is one greater than the reorder level, because the retailers follow base-stock policies. The _STATUS_ variable indicates that the optimization was successful for all observations. The _ALGORITHM_ variable gives information about the algorithm used; namely, a fill rate (FR) service level was used for the retailers, the warehouses follow (s, S) (SS) policies, the retailers follow base-stock (BS) policies, and the gamma distribution (GA) was used for lead-time demand and demand during lead time plus review time for all locations. The remaining variables report several estimated inventory metrics to evaluate the performance of the underlying policy. For information about these variables, see the section “OUT= Data Set” on page 24.

Figure 2.4 Output Data Set policy2

Output Data Set

Obs	sku	warehouse	location	reorder Level	order UpTo Level	avg Inventory
1	A	Raleigh, NC		124	376	67.550
2	A	Raleigh, NC	Atlanta, GA	168	169	66.030
3	A	Raleigh, NC	Baltimore, MD	293	294	98.618
4	A	Raleigh, NC	Charleston, SC	132	133	66.694
5	B	Greensboro, NC		238	745	151.103
6	B	Greensboro, NC	Atlanta, GA	296	297	83.004
7	B	Greensboro, NC	Charleston, SC	217	218	64.683

Obs	avg Backorder	avg Order Freq	avgCost	inventory Ratio	backorder Ratio	turnover
1	26.8844	0.43646	224.864	0.53997	0.21490	1.85197
2	1.9299	0.99984	46.221	2.02547	0.05920	0.49371
3	3.4774	1.00000	69.033	1.59577	0.05627	0.62666
4	1.7799	0.99820	46.686	2.17244	0.05798	0.46031
5	25.7180	0.29107	36.877	1.07700	0.18331	0.92851
6	3.7423	1.00000	8.300	1.21351	0.05471	0.82406
7	3.6626	1.00000	6.468	0.89962	0.05094	1.11158

Obs	fill Rate	ready Rate	_algorithm_	_status_
1	0.79626	0.64136	__-SS-GA	SUCCESSFUL
2	0.94991	0.92813	FR-BS-GA	SUCCESSFUL
3	0.95002	0.91638	FR-BS-GA	SUCCESSFUL
4	0.95101	0.93722	FR-BS-GA	SUCCESSFUL
5	0.83310	0.72480	__-SS-GA	SUCCESSFUL
6	0.94927	0.89959	FR-BS-GA	SUCCESSFUL
7	0.95155	0.88633	FR-BS-GA	SUCCESSFUL

Syntax

The following statements are available in the IRP procedure:

PROC IRP < *options* > ;
HOLDINGCOST *variable* ;
ITEMID *variables* ;
LEADTIME / < *options* > ;
LEADTIMEDEMAND / < *options* > ;
LOCATION *variable* / < *options* > ;
PENALTY / < *options* > ;
POLICYTYPE *variable* ;
REPLENISHMENT / < *options* > ;
REVIEWTIMEDEMAND / < *options* > ;
SERVICE / < *options* > ;

The PROC IRP and ITEMID statements are required. The following sections provide a functional summary of the statements and options you can use in the IRP procedure, then describe the PROC IRP statement, and then describe the other statements in alphabetical order.

Functional Summary

Table 2.2 summarizes the statements and options available for the IRP procedure, classified by function.

Table 2.2 PROC IRP Functional Summary

Description	Statement	Option
Constraints and Policy Specifications		
Maximum ordering frequency	REPLENISHMENT	MAXFREQ=
Minimum order size	REPLENISHMENT	MINSIZE=
Base lot size	REPLENISHMENT	LOTSIZE=
Policy type	POLICYTYPE	
Service type	SERVICE	TYPE=
Service level	SERVICE	LEVEL=
Cost Specifications		
Fixed cost	REPLENISHMENT	FCOST=
Holding cost	HOLDINGCOST	
Penalty cost	PENALTY	COST=
Data Set Specifications		
Input data set	PROC IRP	DATA=
Output data set	PROC IRP	OUT=

Table 2.2 *continued*

Description	Statement	Option
Identifier Variables		
Item ID	ITEMID	
Location	LOCATION	
Lead-Time Specifications		
Lead-time mean	LEADTIME	MEAN=
Lead-time variance	LEADTIME	VARIANCE=
Maximum allowed value of coefficient of variation for lead time	LEADTIME	MAXCOV=
Lead-Time Demand Specifications		
Lead-time demand mean	LEADTIMEDEMAND	MEAN=
Lead-time demand variance	LEADTIMEDEMAND	VARIANCE=
Maximum allowed value of coefficient of variation for lead-time demand	LEADTIMEDEMAND	MAXCOV=
Miscellaneous Options		
Maximum number of items for which input error messages are printed	PROC IRP	MAXMESSAGES=
Estimate of the maximum number of retailer locations	LOCATION	NLOCATIONS=
Optimization Control Specifications		
Type of optimization algorithm	PROC IRP	ALGORITHM=
Choice of probability distribution	PROC IRP	DIST=
Maximum number of iterations	PROC IRP	MAXITER=
Type of policy optimization	PROC IRP	METHOD=
Calculation of optimal policies	PROC IRP	OPTIMAL
Control of scaling of demand and cost parameters	PROC IRP	SCALE=
Granularity of the inventory position distribution for (s, S) policies	REPLENISHMENT	QGRID=
Criterion to determine $S - s$ or Q	REPLENISHMENT	DELTA=
Review-Time Demand Specifications		
Review-time demand mean	REVIEWTIMEDEMAND	MEAN=
Review-time demand variance	REVIEWTIMEDEMAND	VARIANCE=
Maximum allowed value of coefficient of variation for review-time demand	REVIEWTIMEDEMAND	MAXCOV=

PROC IRP Statement

PROC IRP <*options*> **;**

The PROC IRP statement invokes the IRP procedure. You can specify the following *options*:

ALGORITHM=1 | 2

ALG=1 | 2

specifies the type of optimization heuristic to use for single-location inventory systems. This option is ignored when the OPTIMAL option is specified in the PENALTY statement. You can specify the following values:

1 uses an exact optimization algorithm.

2 uses an approximation algorithm.

By default, ALGORITHM=1. For more information, see the section "Policy Algorithm" on page 30.

DATA=*SAS-data-set*

names the SAS data set that contains information about the items to be analyzed. The data set must include the mean and variance of review-time demand, mean replenishment order lead time, per-unit holding cost, fixed replenishment cost, and target service level or back-order penalty cost. The data set can contain other optional variables for use by PROC IRP. For single-location inventory systems, every observation corresponds to an individual inventory item to be analyzed. For two-echelon-distribution systems, every observation corresponds to an inventory item-location pair, and these pairs must be grouped together by item.

The DATA= input data set must be sorted by the variables specified in the ITEMID statement. For more information about the variables in this data set, see the section "Input Data Set" on page 22. If the DATA= option is omitted, the most recently created SAS data set is used.

DIST=AUTO | GAMMA

specifies the type of probability distribution to use for approximating the distributions for both the lead-time demand and the demand during lead time plus review time. You can specify the following values:

AUTO uses the normal distribution whenever appropriate; otherwise uses the gamma distribution.

GAMMA uses the gamma distribution every time.

By default, DIST=AUTO. This option is ignored when the OPTIMAL option is specified in the PENALTY statement. For more information, see the section "Policy Algorithm" on page 30.

MAXITER=*maxiter*

specifies the maximum number of iterations that the heuristic algorithm can use to calculate inventory replenishment policies. By default, MAXITER=100. This option is ignored when the OPTIMAL option is specified in the PENALTY statement.

MAXMESSAGES=*maxmessages*

MAXMSG=*maxmessages*

specifies the maximum number of different items in the DATA= input data set for which input error messages are printed to the SAS log. By default, MAXMESSAGES=100.

METHOD=SERVICE | PENALTY

specifies the optimization method to use for calculating the inventory replenishment policies. You can specify the following values:

SERVICE uses service-level requirements to calculate the replenishment policy.

PENALTY uses back-order penalty costs to calculate the replenishment policy.

By default, METHOD=SERVICE.

OUT=*SAS-data-set*

specifies a name for the output data set that contains inventory replenishment policies, service measure estimates, and other inventory metrics as determined by PROC IRP. This data set also contains all the variables that are specified in the ITEMID statement. Every observation in the DATA= input data set has a corresponding observation in the output data set. For information about the variables in this data set, see the section "OUT= Data Set" on page 24. If this option is omitted, PROC IRP creates a data set and names it according to the DATA*n* naming convention.

HOLDINGCOST Statement

HOLDINGCOST *variable* **;**

HCOST *variable* **;**

The HOLDINGCOST statement identifies the variable in the DATA= input data set that specifies the per-period, per-unit holding cost of each item. (Negative, zero, and missing values for this variable are not permitted.) If this statement is not specified, PROC IRP looks for a variable named HOLDINGCOST. If this variable is not found in the DATA= input data set, PROC IRP halts with an error.

ITEMID Statement

ITEMID *variables* **;**

ID *variables* **;**

SKUID *variables* **;**

The ITEMID statement identifies the variables in the DATA= input data set that specify individual inventory items. For a single-location inventory system, the *variables* primarily identify unique items in the input data set. However, each observation is processed independently, regardless of whether the values of the variables are unique. Thus, you can include any variables that might not necessarily pertain to the descriptions of the items in the list. All *variables* that are specified in this statement are included in the output data set. Therefore, in addition to identifying inventory information (such as SKU), you can also use the ITEMID statement in a single-location inventory system to specify variables that are carried through from the input data set to the output data set. For an illustration, see Example 2.1.

For a two-echelon system, the ITEMID statement specifies the variables in the DATA= input data set that are used to group the observations in the input data set. Each group identifies a single item that is shipped from a warehouse to one or more retailers; each individual observation within a group corresponds to a single warehouse or retailer. The observations within a group are used together to process the group. As in the single-location inventory case, *variables* that are specified in the ITEMID statement are included in the output data set; however, in this case, the variables are used to process observations in groups rather than independently. Thus, you cannot use the ITEMID statement to simply copy variables from the input data set to the output data set (as you can in the single-location inventory system). Instead, you can include a simple DATA step after a call to PROC IRP to merge variables from the input and output data sets.

If the ITEMID statement is not specified, PROC IRP halts with an error. Furthermore, PROC IRP expects the DATA= input data set to be sorted by the variables that are specified in the ITEMID statement. The ITEMID statement behaves much like the BY statement; therefore, you can use options such as DESCENDING and NOTSORTED in the ITEMID statement. For more information about the BY statement, see SAS System documentation.

LEADTIME Statement

LEADTIME / < *options* > **;**

LTIME / < *options* > **;**

The LEADTIME statement identifies the variables in the DATA= input data set that contain the mean and variance of the replenishment order lead time. This information is used to calculate the mean and variance of lead-time demand. The replenishment order lead time should be specified using the same scale as that used for the review periods. This statement is ignored if the LEADTIMEDEMAND statement is specified.

You can specify the following *options*:

MAXCOV=*maxcov*
: specifies the maximum allowed value of the coefficient of variation for replenishment order lead time. Items whose coefficient of variation (ratio of the standard deviation and mean) of lead time is greater than *maxcov* are not processed. By default, MAXCOV=10.

MEAN=*variable*
: identifies the variable in the DATA= input data set that contains the mean of the replenishment order lead time. (Negative, zero, and missing values for this variable are not permitted.) If this option is omitted, PROC IRP looks for a variable named LTMEAN in the DATA= data set. If this variable is not found, PROC IRP halts with an error.

VARIANCE=*variable*

VAR=*variable*
: identifies the variable in the DATA= input data set that contains the variance of the replenishment order lead time. (Negative and missing values for this variable are interpreted as 0.) If this option is omitted, a value of 0 is used for all observations.

LEADTIMEDEMAND Statement

LEADTIMEDEMAND / **MEAN=***variable* **VARIANCE=***variable* < *option* > **;**

LTDEMAND / **MEAN=***variable* **VARIANCE=***variable* < *option* > **;**

The LEADTIMEDEMAND statement identifies the variables in the DATA= input data set that contain the mean and variance of lead-time demand (that is, the amount of demand that occurs during the lead time). The IRP procedure uses the review-time demand and lead-time information to calculate the parameters of lead-time demand. Instead of specifying the parameters of lead time, you can directly specify the mean and variance of lead-time demand by using the LEADTIMEDEMAND statement. This feature is especially useful if lead time is greater than review time and demand is not stationary.

If this statement is specified, the LEADTIME statement is ignored. Because the inventory is periodically reviewed, the lead time in consideration should start after one review period. For an illustration, see Example 2.4.

You must specify the following arguments:

MEAN=*variable*
: identifies the variable in the DATA= input data set that contains the mean of the demand during lead time. (Negative, zero, and missing values for this variable are not permitted.)

VARIANCE=*variable*

VAR=*variable*
: identifies the variable in the DATA= input data set that contains the variance of the demand during lead time. (Negative, zero, and missing values for this variable are not permitted.)

You can specify the following *option*:

MAXCOV=*maxcov*
: specifies the maximum allowed value of the coefficient of variation for lead-time demand. Items whose coefficient of variation (ratio of the standard deviation and mean) of lead-time demand is greater than *maxcov* are not processed. By default, MAXCOV=10.

LOCATION Statement

LOCATION *variable* / < *options* > **;**

LOC *variable* / < *options* > **;**

The LOCATION statement identifies the character variable in the DATA= data set that identifies the retailer locations for the two-echelon-distribution inventory problem. The value of *variable* should be missing if the current observation corresponds to a warehouse. This statement is required for solving two-echelon-distribution inventory problems. If this statement is omitted, each observation is treated as a separate single-location inventory problem.

You can specify the following *option*:

NLOCATIONS=*nlocations*

NLOCS=*nlocations*

specifies an estimate of the maximum number of retailer locations in a single item group for the two-echelon-distribution inventory problem. This option is used for initial memory allocation. By default, NLOCATIONS=50.

PENALTY Statement

PENALTY / < *options* > **;**

The PENALTY statement enables you to specify back-order penalty cost information. This statement is ignored if METHOD=SERVICE in the PROC IRP statement.

You can specify the following *options*:

COST=*variable*

identifies the variable in the DATA= input data set that specifies the per-period, per-unit item penalty cost for backlogged demand. (Negative, zero, and missing values for this variable are not permitted. In addition, values of this variable must be greater than or equal to 1.5 times the value of the variable specified in the HOLDINGCOST statement. This limitation prevents accidental user input errors and guarantees a minimum ready rate of at least 60%.) If METHOD=PENALTY in the PROC IRP statement and this option is not specified, PROC IRP looks for a variable named PENALTYCOST in the DATA= input data set. If this variable is not found, PROC IRP halts with an error.

OPTIMAL

OPT

requests that an optimal policy be calculated. This option is valid only if the LOCATION statement is not specified. By default, PROC IRP uses a heuristic method to calculate nearly optimal policies. For more information, see the section "OPTIMAL Option" on page 33.

SCALE=*scale*

controls the initial scaling of demand and cost parameters for optimal policy calculations. Initial scaling takes place if the calculated mean of demand during lead time plus review time is greater than *scale*. This option is ignored if the OPTIMAL option is not specified. Valid values are between 50 and 10,000. In general, the default scaling is sufficient to produce fast and accurate results. If desired, you can obtain more accuracy at the expense of longer execution time by increasing *scale* (thus decreasing the effective scaling). However, increasing *scale* increases the demand on memory and might result in an error. For more information, see the section "OPTIMAL Option" on page 33. By default, SCALE=100.

POLICYTYPE Statement

POLICYTYPE *variable* **;**

PTYPE *variable* **;**

The POLICYTYPE statement identifies the variable in the DATA= input data set that specifies the type of inventory replenishment policy to be calculated. Table 2.3 lists the valid values of this variable. For more information about policy types, see the section "Replenishment Policies" on page 27.

Table 2.3 Valid Values for the POLICYTYPE Variable

Value	Policy Type
BS	Base-stock policy
SS	(s, S) policy (default)
NQ	(s, nQ) policy, fixed ordering cost for each lot ordered
RQ	(s, nQ) policy, single fixed ordering cost independent of the number of lots ordered

If this statement is not specified, PROC IRP assumes the (s, S) policy for all items in the DATA= input data set.

REPLENISHMENT Statement

REPLENISHMENT */ < options >* **;**

ORDER */ < options >* **;**

REP */ < options >* **;**

You can specify the following *options*:

DELTA=POWER | EOQ

specifies the method used for calculating the difference, $\Delta = S - s$, for (s, S) policies or the base lot size, Q, for (s, nQ) policies. You can specify the following values:

POWER uses a power approximation to determine Δ or Q.

EOQ uses the classic economic order quantity to determine Δ or Q.

For more information, see the section "Policy Algorithm" on page 30. By default, DELTA=POWER.

FCOST=*variable*

identifies the variable in the DATA= input data set that specifies the fixed ordering cost of placing a replenishment order. (Negative and missing values for this variable are interpreted as 0.) If this option is not specified, PROC IRP looks for a variable named FIXEDCOST in the DATA= input data set. If this variable is not found, PROC IRP halts with an error.

LOTSIZE=_variable_

identifies the variable in the DATA= input data set that specifies the difference, $\Delta = S - s$, for (s, S) policies or the base lot size, Q, for (s, nQ) policies. (Negative, zero, and missing values for this variable are ignored.)

MAXFREQ=_variable_

identifies the variable in the DATA= input data set that contains the maximum allowable average ordering frequency. In practice, the fixed cost of placing an order can be difficult to estimate; therefore, this variable enables you to limit the frequency with which orders are placed. (Negative, zero, and missing values for this variable are ignored. Furthermore, for (s, S) policies, the value cannot be less then 1/*qgrid*, where *qgrid* is the value specified by the QGRID= option, or PROC IRP halts with an error.)

MINSIZE=_variable_

identifies the variable in the DATA= input data set that contains the minimum allowable replenishment order size. (Negative and missing values for this variable are ignored, with the exception of –1. A value of –1 is a special flag and sets the minimum order size to 1.5 times the average one-period demand.) If this option is omitted, a value of 0 is used for all observations.

QGRID=_qgrid_

identifies the granularity of the inventory position distribution for (s, S) policies. This option is used only for the heuristic algorithms and is ignored when the OPTIMAL option is specified in the PENALTY statement. Valid values are between 5 and 100. By default, QGRID=10, which is appropriate for most situations. However, specifying a value greater than the default might result in better accuracy (at the expense of computation time).

REVIEWTIMEDEMAND Statement

REVIEWTIMEDEMAND / < *options* > **;**

RTDEMAND / < *options* > **;**

The REVIEWTIMEDEMAND statement identifies the variables in the DATA= input data set that contain the mean and variance of the review-time demand. When the REVIEWTIMEDEMAND statement is specified, demand during the review periods is assumed to be stationary and independent.

You can specify the following *options*:

MAXCOV=_maxcov_

specifies the maximum allowed value of the coefficient of variation for review-time demand. Items whose coefficient of variation (ratio of the standard deviation and mean) of review-time demand is greater than *maxcov* are not processed. By default, MAXCOV=10.

MEAN=_variable_

identifies the variable in the DATA= input data set that contains the mean of the demand during a single inventory review period. (Missing values and values less than 1 for this variable are not permitted. However, the mean of review-time demand at the warehouse in the two-echelon-distribution problem can be set to missing to instruct PROC IRP to automatically calculate the mean and variance of demand at the warehouse as the sum of the means and variances of demand at the retailer locations.) If this option is omitted, PROC IRP looks for a variable named RTDMEAN in the DATA= input data set. If this variable is not found, PROC IRP halts with an error.

VARIANCE=*variable*

VAR=*variable*

identifies the variable in the DATA= input data set that contains the variance of the demand during a single inventory review period. (Negative and missing values for this variable are interpreted as 0.) If this statement is omitted, PROC IRP looks for a variable named RTDVAR in the DATA= input data set. If this variable is not found, PROC IRP halts with an error.

SERVICE Statement

SERVICE / < *options* > **;**

The SERVICE statement identifies the variables in the DATA= input data set that specify the type and the desired level of the service measure to be used by the inventory policy algorithm. This statement is ignored if METHOD=PENALTY in the PROC IRP statement.

You can specify the following *options*:

LEVEL=*variable*

identifies the variable in the DATA= input data set that specifies the desired service level for the service measure that is specified in the TYPE= option. (Common ranges of service level are [0.80, 0.99] for fill rate and ready rate and [0.01, 0.20] for back-order ratio. Valid values for fill rate and ready rate are between 0.600 and 0.999 and for back-order ratio are between 0.001 and 0.400.) If METHOD=SERVICE in the PROC IRP statement and this option is not specified, PROC IRP looks for a variable named SERVICELEVEL in the DATA= input data set. If this variable is not found, PROC IRP halts with an error.

TYPE=*variable*

identifies the variable in the DATA= input data set that specifies the type of service measure to be used by the inventory replenishment algorithm. Only one service measure per item can be specified in a single procedure invocation. Table 2.4 lists the valid values for this variable.

Table 2.4 Valid Values for the TYPE= Variable

Value	Service Measure
FR	Fill rate (default)
RR	Ready rate
BR	Back-order ratio

If this option is not specified, PROC IRP assumes a fill rate service measure for all items in the DATA= input data set.

Details

This section provides detailed information about the use of the IRP procedure. Subsections describe different aspects of the procedure.

Input Data Set

PROC IRP uses data from the DATA= input data set, where key variable names identify the appropriate information. Table 2.5 lists all the variables associated with the input data set and their interpretation by the IRP procedure. The variables are grouped according to the statement with which they are specified. In addition, the fourth column shows variables that have default names and do not need to be specified in any of the procedure statements.

Table 2.5 PROC IRP Input Data Set and Associated Variables

Statement	Option That Specifies Variable Name	Variable Interpretation	Default Variable Name
HOLDINGCOST	HOLDINGCOST	Holding cost	HOLDINGCOST
ITEMID	ITEMID	Item identifier	
LEADTIME	MEAN=	Lead time mean	LTMEAN
	VARIANCE=	Lead time variance	
LEADTIMEDEMAND	MEAN=	Lead-time demand mean	
	VARIANCE=	Lead-time demand variance	
LOCATION	LOCATION	Retailer location identifier	
PENALTY	COST=	Back-order penalty cost	PENALTYCOST
POLICYTYPE	POLICYTYPE	Policy type	
REPLENISHMENT	FCOST=	Fixed ordering cost	FIXEDCOST
	LOTSIZE=	Base lot size	
	MAXFREQ=	Maximum ordering frequency	
	MINSIZE=	Minimum order size	
REVIEWTIMEDEMAND	MEAN=	Review-time demand mean	RTDMEAN
	VARIANCE=	Review-time demand variance	RTDVAR
SERVICE	LEVEL=	Desired service level	SERVICELEVEL
	TYPE=	Service measure type	

Missing Values in the Input Data Set

Table 2.6 summarizes the treatment of missing values for variables in the DATA= input data set.

Table 2.6 Treatment of Missing Values in the IRP Procedure

Statement	Option That Specifies Variable Name	Action Taken
HOLDINGCOST	HOLDINGCOST	Input error: procedure moves to processing of next ITEMID group
LEADTIME	MEAN=	Input error: procedure moves to processing of next ITEMID group
	VARIANCE=	Value is assumed to be 0
LEADTIMEDEMAND	MEAN=	Input error: procedure moves to processing of next ITEMID group
	VARIANCE=	Input error: procedure moves to processing of next ITEMID group
LOCATION	LOCATION	Current observation is assumed to define a warehouse
PENALTY	COST=	Input error: procedure moves to processing of next ITEMID group if METHOD= PENALTY; value is ignored if METHOD=SERVICE
POLICYTYPE	POLICYTYPE	Value is assumed to be SS
REPLENISHMENT	FCOST=	Value is assumed to be 0
	LOTSIZE=	Value is ignored
	MAXFREQ=	Value is ignored
	MINSIZE=	Value is assumed to be 0
REVIEWTIMEDEMAND	MEAN=	Input error (unless the value of the LOCATION variable is also missing): procedure moves to processing of next ITEMID group
	VARIANCE=	Value is assumed to be 0 (or the sum of other ITEMID group values if the value of the LOCATION variable is missing)
SERVICE	LEVEL=	Input error: procedure moves to processing of next ITEMID group if METHOD=SERVICE; value is ignored if METHOD=PENALTY
	TYPE=	Value is assumed to be FR

OUT= Data Set

The OUT= data set contains the inventory replenishment policies for the items identified in the DATA= input data set. The OUT= data set contains one observation for each observation in the DATA= input data set. If an error is encountered while PROC IRP processes an observation, information about the error is written to the OUT= data set.

Definitions of Variables in the OUT= Data Set

Each observation in the OUT= data set is associated with an individual inventory item (SKU). The variables that are specified in the ITEMID statement are copied to the OUT= data set. The following variables are also added to the OUT= data set:

AVGBACKORDER
contains the estimated average back orders for the calculated inventory replenishment policy. Average back orders are the average number of cumulative back orders in a review period. This estimate loses accuracy if the lead time is not an integer multiple of the review period or if the variance of lead time is high.

AVGCOST
contains the estimated average cost per period for the calculated inventory replenishment policy. Average cost is the average cost (including holding, ordering, and back-order penalty costs) incurred per review period.

AVGINVENTORY
contains the estimated average inventory for the calculated inventory replenishment policy. Average inventory is the average on-hand inventory at the end of a review period. This estimate loses accuracy if the lead time is not an integer multiple of the review period or if the variance of lead time is high.

AVGORDERFREQ
contains the estimated average ordering frequency for the calculated inventory replenishment policy. Average ordering frequency is the average number of replenishment orders placed per review period.

BACKORDERRATIO
contains the estimated back-order ratio for the calculated inventory replenishment policy. The back-order ratio is the average number of back orders divided by the average demand.

FILLRATE
contains the estimated fill rate for the calculated inventory replenishment policy. Fill rate is the fraction of demand that is satisfied from on-hand inventory. If the OPTIMAL option is specified in the PENALTY statement, the FILLRATE variable is not added to the OUT= data set.

INVENTORYRATIO
contains the estimated inventory ratio for the calculated inventory replenishment policy. The inventory ratio is the average inventory divided by the average demand.

ORDERUPTOLEVEL
specifies the order-up-to level, S, for (s, S) policies or the sum of the reorder level and the base lot size, $s + Q$, for (s, nQ) policies.

READYRATE

contains the estimated ready rate for the calculated inventory replenishment policy. The ready rate is the probability of no stockout in a review time period.

REORDERLEVEL

specifies the reorder level, s. The reorder level is the inventory level at which a replenishment order should be placed.

TURNOVER

contains the estimated turnover for the calculated inventory replenishment policy. Turnover is the average demand divided by the average inventory. The value of this variable is set to missing if the estimated average inventory is 0.

ALGORITHM

indicates which algorithm was used to calculate the inventory replenishment policy. The value of the _ALGORITHM_ variable is in the form XX-YY-ZZ, where XX indicates the type of optimization used, YY indicates the type of policy calculated, and ZZ indicates the approximation used for the distribution for lead-time demand and for demand during lead time plus review time. Table 2.7 shows the possible values of the _ALGORITHM_ variable.

Table 2.7 Possible Values of the _ALGORITHM_ Variable

String	Value	Description
XX	BR	Back-order ratio
	FR	Fill rate
	PC	Penalty cost
	RR	Ready rate
YY	BS	$(S-1, S)$ base-stock policy
	NQ	(s, nQ) policy, fixed ordering cost for each lot ordered
	RQ	(s, nQ) policy, single fixed ordering cost independent of the number of lots ordered
	SS	(s, S) policy (or (s, nQ, S) policy if a base lot size Q is specified)
ZZ	GA	Gamma distribution
	NO	Normal distribution

The ZZ portion of this variable has a slightly different format when the OPTIMAL option is specified in the PENALTY statement. For more information, see the section “OPTIMAL Option” on page 33.

For two-echelon-distribution systems, the XX portion of this variable has the value ‘__’ when the current value of the LOCATION variable defines a warehouse, because no service constraints or penalty costs are applied at the warehouse.

SCALE

contains the value used to scale the demand and cost parameters during policy calculations. If scaling is performed (that is, if the value of _SCALE_ is greater than 1), all values that are written to the OUT= data set are in original units. This variable is added to the OUT= data set only when the OPTIMAL option is specified in the PENALTY statement. For more information about scaling, see the section "OPTIMAL Option" on page 33.

STATUS

contains the completion status of the inventory replenishment algorithm. Table 2.8 shows the possible values of the _STATUS_ variable.

Table 2.8 Possible Values of the _STATUS_ Variable

Value	Explanation
SUCCESSFUL	Successful completion
INVD_VALUE	Invalid value in the DATA= input data set
MAX_ITER	Maximum number of iterations reached
INSUF_MEM	Insufficient memory
BAD_DATA	Numerical or scaling problem encountered

Error Processing

For single-location inventory systems, PROC IRP processes each item (observation) individually. If an error occurs, PROC IRP stops processing the current item and writes information about the type of error to the _STATUS_ variable in the OUT= data set. Execution resumes with the next item.

For two-echelon-distribution systems, PROC IRP processes items in groups (multiple observations) that represent the warehouse and retailer locations. If an error is detected for any of the corresponding observations, PROC IRP stops processing the current item group and the type of error is noted in the _STATUS_ variable for all items in the group. Execution resumes with the next item group.

At procedure termination, the value of the macro variable, _IRPIRP_, is set appropriately to reflect the fact that errors were encountered during execution.

Macro Variable _IRPIRP_

PROC IRP defines a macro variable named _IRPIRP_ . This variable is set at procedure termination and contains a character string that indicates the status of the procedure. The form of the _IRPIRP_ character string is STATUS=*status* NSUCCESS=*nsuccess* NFAIL=*nfail*, where *nsuccess* is the number of items successfully processed, *nfail* is the number of items for which the policy calculation has failed, and *status* can be one of the following:

- SUCCESSFUL (indicates successful completion of the procedure)
- RUNTIME_ERROR (indicates that policy calculations failed for at least one item or item group in the DATA= input data set)

- SYNTAX_ERROR (indicates failure caused by a procedure syntax error)
- MEMORY_ERROR (indicates failure during procedure initialization or data input parsing caused by insufficient memory)

This information can be used when PROC IRP is one step in a larger program that needs to determine whether the procedure terminated successfully or not. Because _IRPIRP_ is a standard SAS macro variable, it can be used in the ways that all macro variables can be used.

Replenishment Policies

PROC IRP calculates four types of replenishment policies, based on the specified policy type:

- **SS**=(s, S) policy: When the inventory position falls to or below the reorder level, s, an order is placed so as to raise the inventory position to the order-up-to level, S. In other words, if the inventory position is y and $y \leq s$, then an order of size $S - y$ is placed. The (s, S) policy is sometimes referred to as the *min-max policy*. Note that the size of the replenishment order is always greater than or equal to $S - s$.

- **BS**=(s, S) policy with $S = s + 1$ (base-stock policy): When the inventory position falls to or below the reorder level, s, an order is placed so as to raise the inventory position to the order-up-to level, S. When $S = s + 1$, the (s, S) policy is called a *base-stock policy*. (A base-stock policy is also called an "order-up-to policy," "one-to-one replenishment policy," or "installation stock policy.")

- **NQ**=(s, nQ) policy when you have a fixed ordering cost for each lot ordered: You incur a fixed ordering cost for each lot ordered. When the inventory position falls to or below the reorder level, s, an order is placed to bring the inventory position to just above s. The size of this order is a multiple of the base lot size, Q.

  In other words, if the inventory position is y and $y \leq s$, then an order of size nQ is placed, where n is the smallest integer such that $y + nQ > s$. In this case, both s and Q are decision variables; you can use the LOTSIZE= option in the REPLENISHMENT statement if Q is to be a previously specified value rather than a decision variable. When $Q = 1$, the (s, nQ) policy becomes a base-stock policy.

- **RQ**=(s, nQ) policy when you incur a single fixed ordering cost: You incur a single fixed ordering cost independent of the number of lots ordered. When the inventory position falls to or below the reorder level, s, an order is placed to bring the inventory position to just above s. The size of this order is a multiple of the base lot size, Q.

  In other words, if the inventory position is y and $y \leq s$, then an order of size nQ is placed, where n is the smallest integer such that $y + nQ > s$. In this case, both s and Q are decision variables; you can use the LOTSIZE= option in the REPLENISHMENT statement if Q is to be a previously specified value rather than a decision variable. When $Q = 1$, the (s, nQ) policy becomes a base-stock policy.

For single-location inventory systems under standard assumptions (independent customer demands, full back-ordering of unfulfilled demand, fixed replenishment ordering costs, linear inventory holding costs, and linear back-order penalty costs), (s, nQ) policies are known to be suboptimal and (s, S) policies are known to be optimal. Although (s, S) policies are optimal, the restricted order size under an (s, nQ) policy might better facilitate easy packaging, transportation, and coordination in some situations.

Inventory Costs

Because the objective of inventory planning is usually to minimize costs, the assumptions about the cost structure are important. There are three types of costs:

- *Ordering cost* is the cost incurred every time a replenishment order is placed. This fixed cost includes the expense of processing the order and is usually independent of the size of the order.
- *Holding cost* is the cost of carrying inventory and might include the opportunity cost of money invested, the expenses of running a warehouse, handling and counting costs, the costs of special storage requirements, deterioration of stock, damage, theft, obsolescence, insurance, and taxes. The most common convention is to specify holding cost (per-period, per-unit) as a fixed percentage of the unit cost of the item. This cost is then applied to the average inventory.
- *Penalty* (*back-ordering or shortage*) *cost* is the cost incurred when a stockout occurs. This cost might include the cost of emergency shipments, cost of substitution of a less profitable item, or cost of lost goodwill. For example, will the customer ever return? Will the customer's colleagues be told of the unsatisfactory service? The most common convention is to specify penalty cost as per-period, per-unit and then apply it to the average number of back orders.

In practice, it is often difficult to estimate the ordering (replenishment) cost and the penalty cost. As a result, practitioners often put restrictions on the ordering frequency rather than estimate the cost of ordering. Likewise, specific target levels for service measures can be substituted for the penalty cost.

Service Measures

Service measures are often used to evaluate the effectiveness of an inventory replenishment policy. You can influence policy calculations by imposing desired service-level requirements. PROC IRP supports the use of three different service constraints:

- *Fill rate* is the fraction of demand that is satisfied directly from on-hand inventory. Fill rate is one of the most frequently used service measures in practice. You can set a minimum fill rate as a service constraint.
- *Ready rate* is the probability of no stockout in a review period. You can set a minimum ready rate as a service constraint.
- *Back-order ratio* is the average number of back orders divided by the average demand. You can set a maximum back-order ratio as a service constraint.

These service constraints provide different ways of penalizing back orders. When fill rate is used as a service measure, the focus is only on the number of back orders, whereas when back-order ratio is a service measure, the focus is on both the number and the length of back orders. When ready rate is used as a service measure, the focus is not on the number or length of back orders, but on whether or not a stockout occurs.

Setting a high target service level might result in high inventory levels, which can be very costly if demand is intermittent (slow-moving). In these cases, estimating penalty costs and performing a cost optimization might be preferred.

PROC IRP reports several other measures to evaluate the performance of a policy:

- *Average ordering frequency* is the number of replenishment orders placed per review period. You can set a limit on the average ordering frequency.
- *Average inventory* is the average on-hand inventory at the end of a review period.
- *Average back order* is the average amount of outstanding back-ordered demand in a review period.
- *Inventory ratio* is the average inventory divided by the average demand.
- *Turnover* is the average demand divided by the average inventory.
- *Average cost* is the average cost (holding and replenishment) incurred per period. If back-order penalty costs are present, these are included as well.

Lost Sales

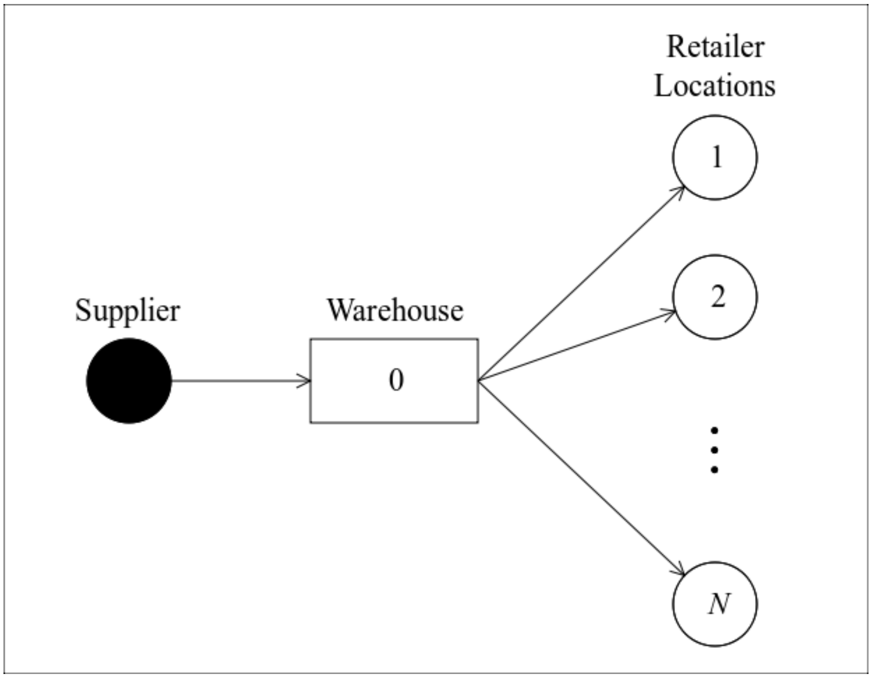
A *lost-sales* inventory system enables unsatisfied demand to be lost rather than back-ordered. For an (s, S) policy, this system can be approximated by using the fill rate service measure with some slight modifications (Tijms and Groenevelt 1984).

Let β_l represent the fraction of satisfied demand in the lost-sales case. Therefore, $1 - \beta_l$ represents the fraction of demand that is lost. The reorder and order-up-to levels for the lost-sales inventory system are approximately the same as those in a back-ordering inventory system that has a target fill rate service level specified as $\beta_f = 2 - 1/\beta_l$. This approximation should be used only when β_l is close to 1.

Two-Echelon-Distribution Inventory System

PROC IRP can find nearly optimal policies for two-echelon-distribution inventory systems that have different service constraints on multiple retailer locations. A two-echelon-distribution inventory system consists of a single warehouse and *N* retailer locations. The retailer locations pull items from the warehouse, and the items are supplied to the warehouse by an exogenous supplier. Figure 2.5 shows a two-echelon-distribution inventory system, where node *0* designates the warehouse and nodes *1* through *N* designate the retailer locations.

Figure 2.5 Two-Echelon-Distribution Inventory System



Retailer locations place replenishment orders at the warehouse according to a base-stock policy. The replenishment cost of retailer locations is negligible or constant. There is a lead time L_i from the warehouse to retailer i. In addition, there is a lead time L_0 from the outside supplier to the warehouse. If the warehouse has sufficient inventory on hand, it immediately dispatches the order, so that the order arrives at the retailer location after the appropriate lead time. If the warehouse has some inventory on hand but not enough to fill the entire order, it partially fills the order with the on-hand inventory and back-orders the rest. If the warehouse has no inventory on hand when a retailer location places an order, the warehouse back-orders the entire order. Note that the average lead time realized at retailer location i is greater than L_i, because the retailer locations have to wait longer if the warehouse is out of stock. All orders that the warehouse receives have the same priority. The warehouse follows an (s, S) or (s, nQ) policy and incurs a fixed replenishment cost every time it places an order from the outside supplier. The retailer locations can have different demand patterns and service constraints. If the penalty costs on back orders at the retailer locations are known, the total system cost incurred per period is minimized.

Policy Algorithm

Single-Location Inventory Systems

When the IRP procedure is used to calculate replenishment policies for single-location inventory systems, the underlying assumptions of the optimization model are as follows:

- The holding and stockout costs are linear.
- The probability that replenishment orders cross in time or arrive simultaneously is negligible.

- The stock on hand just after arrival of a replenishment order is positive except for a negligible probability.
- The review-period demand is independent and identically distributed (stationary). If demand is nonstationary, PROC IRP can still find nearly optimal policies by using the LEADTIMEDEMAND statement. For an illustration, see Example 2.4.

Let

$$
\begin{aligned}
D_R &= \text{demand during review time} \\
D_L &= \text{demand during lead time} \\
L &= \text{lead time (in number of review periods)} \\
D_{LR} &= \text{demand during lead time plus review time} \\
OF &= \text{ordering frequency per period} \\
I &= \text{on-hand inventory at the end of a period} \\
B &= \text{outstanding back orders in a period} \\
K &= \text{fixed cost of replenishment} \\
h &= \text{holding cost per period} \\
p &= \text{penalty cost per period}
\end{aligned}
$$

PROC IRP supports two different methods of solving single-location inventory problems. When METHOD=SERVICE, PROC IRP uses a service-level requirement to constrain the optimization. Alternatively, when METHOD=PENALTY, PROC IRP uses back-order penalty costs to drive the optimization.

By default, PROC IRP uses a heuristic algorithm to calculate nearly optimal policies. If you use the penalty cost method, you can specify the OPTIMAL option in the PENALTY statement to request that PROC IRP calculate optimal policies.

The type of policy that the IRP procedure calculates is determined by the value of the POLICYTYPE variable. For more information, see the section “POLICYTYPE Statement” on page 19.

Service Constraint Method

If METHOD=SERVICE, PROC IRP finds nearly optimal policies in which the replenishment and holding costs are minimized subject to a service-level constraint. PROC IRP calculates the policy in three steps:

1. It calculates the mean and variance of D_L and D_{LR} (unless they are specified in the LEADTIMEDEMAND statement).
2. The algorithm finds $\Delta = S - s$ (the gap between S and s for (s, S) policies) if the value of the POLICYTYPE variable is SS, or it finds $\Delta = Q$ (the base lot size for (s, nQ) policies) if the value of the POLICYTYPE variable is RQ or NQ.

   If the fixed replenishment cost, K, is specified by the FCOST= variable, Δ is determined according to the specification of the DELTA= option. If DELTA=EOQ, Δ is set to the classic economic order quantity (EOQ). If DELTA=POWER, a power approximation similar to the one in Ehrhardt and Mosier 1984 is used to determine Δ.

If the fixed replenishment cost, K, is not known, or if there is either a constraint on Δ (specified in the MINSIZE= variable) or a constraint on the ordering frequency specified in the MAXFREQ= variable, Δ is adjusted so that these constraints are met. If a base lot size, Q, is specified in the LOTSIZE= variable, Δ is set to Q and all other constraints are ignored.

3. The reorder level, s, is found such that the user-specified service type and desired service level are met.

The optimization algorithm that is applied depends on the ALGORITHM= option. If it is specified as 1, the calculations are exact. If it is specified as 2, an approximation algorithm is used.

The approximation is fast, works well for large Δ ($\Delta \geq 1.5 \times E(D_R)$), and leads to nearly optimal solutions. If Δ is small, policy parameters are modified. For (s, S) policies, the reorder level, s, and order-up-to level, S, are determined as

$$s = \begin{cases} S_b, & S_b < s_p \\ s_p, & s_p \leq S_b \leq S_p \\ S_b - \Delta, & S_b > S_p \end{cases}$$

$$S = S_b$$

where (s_p, S_p) is the policy that is found, assuming Δ is large and S_b is the base stock level for the same problem. If there are constraints on the order size or the ordering frequency, these are also taken into account. For (s, nQ) policies, the reorder level, s, and base lot size, Q, are determined as

$$s = S_b - \Delta/2$$

$$Q = \Delta$$

where S_b is the base stock level for the same problem.

A suitable distribution must be chosen to represent distributions for both lead-time demand and demand during lead time plus review time. In practice, the normal distribution is widely used to approximate these distributions. However, this choice can lead to very poor results if the coefficients of variation are not small. To overcome this drawback of the normal distribution, PROC IRP uses the gamma distribution if the coefficient of variation of demand during lead time plus review time is greater than 0.5. In both cases, a two-moment approximation is used.

Penalty Cost Method

If METHOD=PENALTY, PROC IRP finds nearly optimal policies in which the replenishment, holding, and back-order penalty costs are minimized. The policy calculation is the same as outlined in the section "Service Constraint Method" on page 31, except for the final step.

In the final step, the reorder level, s, is found such that the average cost per period

$$C(s, \Delta) = K\,E(OF) + h\,E(I) + p\,E(B)$$

is minimized. The choice of distribution that is used to represent both lead-time demand and demand during lead time plus review time is the same as described in the section "Service Constraint Method" on page 31.

Note that this heuristic finds Δ and s sequentially. You can specify the OPTIMAL option to find true optimal policies for single-location systems, in which Δ and s are jointly optimized. For more information, see the section "OPTIMAL Option" on page 33.

Base-Stock Policies

If the value of the POLICYTYPE variable is BS, or if there is no cost of ordering ($K = 0$) and there are no constraints on the order size or the ordering frequency, a base-stock policy is calculated. The policy calculation is similar to what is outlined in the section "Service Constraint Method" on page 31. The difference is that step 2 is skipped because $\Delta = 1$ for base-stock policies.

OPTIMAL Option

If the OPTIMAL option in the PENALTY statement is specified, PROC IRP finds optimal (s, S) or (s, nQ) policies for single-location inventory systems. (For more information, see Zheng and Federgruen 1992 and Zheng and Chen 1992.) The decision variables are s and S for (s, S) policies and s and Q for (s, nQ) policies. In this case, the variables that are specified by the LOTSIZE=, MAXFREQ=, and MINSIZE= options in the REPLENISHMENT statement are ignored. The algorithm that PROC IRP uses when the OPTIMAL option is specified is slower than the heuristic algorithm. Note that the OPTIMAL option is not available for two-echelon-distribution inventory systems.

Define the following notation:

$C_{SS}(s, S)$	=	average cost of an (s, S) policy (when the value of the POLICYTYPE variable is SS)
$C_{NQ}(s, Q)$	=	average cost of an (s, nQ) policy when the fixed cost K is incurred for each lot Q ordered (when the value of the POLICYTYPE variable is NQ)
$C_{RQ}(s, Q)$	=	average cost of an (s, nQ) policy when the fixed cost K is incurred independent of the number of lots ordered (when the value of the POLICYTYPE variable is RQ)

In each instance, PROC IRP finds optimal values of the decision variables s^*, S^*, and Q^* such that the average cost per period is minimized:

$$\begin{aligned} C_{SS}(s^*, S^*) &= \min_{\forall s,S} C_{SS}(s, S) \\ C_{NQ}(s^*, Q^*) &= \min_{\forall s,Q} C_{NQ}(s, Q) \\ C_{RQ}(s^*, Q^*) &= \min_{\forall s,Q} C_{RQ}(s, Q) \end{aligned}$$

Each optimal policy is optimal within its own class. Note that (s, S) policies are optimal among *all* classes of policies for single-location inventory systems:

$$C_{SS}(s^*, S^*) \leq C_{RQ}(s^*, Q^*) \leq C_{NQ}(s^*, Q^*)$$

Suitable distributions must be chosen to represent lead-time demand and review-time demand. These distributions are assumed to be discrete. If the variance is greater than the mean, the distribution under consideration is approximated by a negative binomial distribution. If the variance is less than or equal to the mean, a shifted Poisson distribution is used. The negative binomial and shifted Poisson distributions are fit such that the resulting mean and variance match the mean and variance of the original distribution. The chosen distributions are indicated by a B or P in the ZZ part of the _ALGORITHM_ variable, where the first Z indicates the approximation used for lead-time demand plus review-time demand distribution, and the second Z indicates the approximation used for the review-time demand distribution. While choosing an appropriate distribution, the algorithm might choose a deterministic distribution (a fixed number) to represent these distributions if the variance is close to zero or considerably smaller than the mean. In that case, this number matches the mean of the estimated distribution and is indicated by a D (for deterministic) in the _ALGORITHM_ variable. If the chosen policy is NQ, the review-time demand distribution does not play a role in the optimization algorithm. This is indicated by a '_' in the _ALGORITHM_ variable.

The OUT= data set contains a new variable named _SCALE_, which gives the value used to scale the demand and cost parameters. Initial scaling takes place if the calculated mean of demand during lead time plus review time is greater than the value specified in the SCALE= option. PROC IRP might perform further scaling to obtain a suitable fit to the shifted Poisson or negative binomial distribution. If the procedure is unable to find a suitable fit, it stops processing the current item and writes the value "BAD_DATA" to the _STATUS_ variable. Increasing the value of the SCALE= option might resolve this issue.

Both the magnitude of demand and cost parameters affect the amount of memory required for the algorithm to calculate policies. In some cases, if insufficient scaling is performed, PROC IRP might run out of memory. If PROC IRP runs out of memory, it stops processing the current item and writes the value "INSUF_MEM" to the _STATUS_ variable. Usually, decreasing the value of the SCALE= option corrects this problem. Note that a smaller value for the SCALE= option results in scaling by an equal or larger value.

Two-Echelon-Distribution Inventory Systems

When the IRP procedure is used to calculate replenishment policies for two-echelon-distribution inventory systems, the underlying assumptions of the optimization model are the same as in single-location inventory systems. Let

$$
\begin{aligned}
OF_0 &= \text{ordering frequency per period at the warehouse} \\
S_0 &= \text{order-up-to level at the warehouse} \\
s_i &= \text{reorder level at location } i = 0, \ldots, N \\
\mu_i &= \text{review-time demand at location } i = 0, \ldots, N \\
K_i &= \text{fixed cost of replenishment at location } i = 0, \ldots, N \\
h_i &= \text{holding cost per period at location } i = 0, \ldots, N \\
p_i &= \text{penalty cost per period at location } i = 1, \ldots, N \\
I_i &= \text{on-hand inventory at end of period at location } i = 0, \ldots, N \\
B_i &= \text{outstanding back orders in a period at location } i = 0, \ldots, N
\end{aligned}
$$

Location $i = 0$ refers to the warehouse.

PROC IRP supports two different methods for solving two-echelon-distribution inventory problems. When METHOD=SERVICE, PROC IRP uses a service-level requirement to constrain the optimization. Alternatively, when METHOD=PENALTY, PROC IRP uses back-order penalty costs to drive the optimization.

For two-echelon-distribution inventory systems, PROC IRP calculates base-stock policies for each retailer location. The type of policy for the warehouse is determined by the value of the POLICYTYPE variable.

Service Constraint Method

If METHOD=SERVICE, PROC IRP finds nearly optimal policies in which the replenishment and holding costs are minimized subject to service-level constraints on the retailer locations. The policy calculation is done in three steps:

1. The mean and variance of review-time demand at the warehouse are calculated as the sum of the means and the sum of the variances of review-time demand at the retailer locations, respectively. However, if the mean and variance of review-time demand at the warehouse are explicitly specified in the DATA= input data set, those values are used instead.

   A collaborative forecast (for all retailer locations) might yield a better prediction of the variance of review-time demand at the warehouse than the sum of the variances at the retailer locations. Also, specifying a value for review-time demand at the warehouse that is significantly different from the sum of the means of review-time demand at the retailer locations might cause numerical problems.

   Next, the mean and variance of lead-time demand at the warehouse are calculated.

2. This step is the same as for the single-location inventory problem in the section "Service Constraint Method" on page 31; $\Delta_0 = S_0 - s_0$ is calculated for the warehouse. In order to increase the performance of the algorithm for (s, S) policies, a base-stock policy at the warehouse is assumed if the calculated Δ_0 is less than $0.75\mu_0$ and there are no constraints on the order size.

3. The average cost per period incurred by the system is given as

   $$C(s_0, \Delta_0, s_1, s_2, \ldots, s_N) = K_0 E(OF_0) + \sum_{i=0}^{N} h_i E(I_i)$$

   In this final step, the cost function is minimized subject to the service-level constraint at each retailer location. The decision variables are $s_i,\ i = 0, 1, \ldots, N$. Note that $\Delta_0 = S_0 - s_0$ is calculated and fixed in step 2.

   Each retailer location might have a constraint on fill rate, ready rate, or back-order ratio.

As with the single-location inventory problem, a distribution needs to be chosen to represent the lead-time demand and the demand during lead time plus review time at the warehouse and the retailer locations. PROC IRP uses the gamma distribution to represent these demand distributions. (See the section "Service Constraint Method" on page 31.)

Note that PROC IRP ignores any information about service levels, service types, and penalty costs at the warehouse, because the back orders at the warehouse are treated implicitly. Similarly, any information about policy type, fixed ordering costs, fixed lot size, minimum order size, and maximum ordering frequency is ignored at the retailers, because they follow a base-stock policy.

Penalty Cost Method

If METHOD=PENALTY, PROC IRP finds nearly optimal policies in which the replenishment, holding, and back-order penalty costs are minimized. The policy calculation is the same as the one outlined in the section "Service Constraint Method" on page 34 except for the final step.

In the final step, the cost function,

$$C(s_0, \Delta_0, s_1, s_2, \ldots, s_N) = K_0 E(OF_0) + \sum_{i=0}^{N} h_i E(I_i) + \sum_{i=1}^{N} p_i E(B_i)$$

is minimized. The decision variables are again $s_i,\ i = 0, 1, \ldots, N$.

Note that this function does not penalize back orders at the warehouse directly (there is no $p_0 E(B_0)$ component). This is justified because customer transactions occur only at the retailer locations. Back orders at the warehouse translate to poor performance at the retailer locations by increasing the replenishment order lead time.

Retailer Location Replenishment Order Lead Time

The replenishment order lead time from the warehouse to any retailer location is equal to the shipping and handling time as long as the warehouse has the necessary quantity in stock. In the event of shortages at the warehouse, the retailer location has to wait for an extra amount of time, which depends on the time required to replenish the warehouse from a supplier. This extra *warehouse wait* is a function of the warehouse reorder and order-up-to levels. This causes the lead-time demand process at a retailer location to depend on both the retailer base-stock level (s_i) and the warehouse reorder and order-up-to levels (s_0 and S_0). PROC IRP estimates the mean and variance of the warehouse wait by using techniques similar to those described in Matta and Sinha 1995.

Examples

This section illustrates how you can use PROC IRP to calculate inventory replenishment policies. Example 2.1 through Example 2.5 focus on a single-location inventory system, and Example 2.6 and Example 2.7 focus on a two-echelon-distribution inventory system.

Example 2.1: Single-Location System: Service-Level Heuristic

This single-location inventory example uses service-level heuristics to calculate inventory replenishment policies. The retailer purchases finished goods from its suppliers and sells them to its customers. There are 10 items to be considered, identified by the SKU variable. Estimates of the holding and fixed costs, mean and variance of the lead time, and mean and variance of the review-time demand are shown in Table 2.9. The missing value for Fixed Cost in the last observation indicates that the fixed cost for S10 is difficult to estimate; a maximum order frequency is placed on this item to account for this difficulty.

Table 2.9 Data Estimates for Single-Location System

		Holding	Fixed	Lead Time		Review-Time Demand	
SKU	**Supplier**	**Cost**	**Cost**	**Mean**	**Variance**	**Mean**	**Variance**
S01	ABC Company	0.78	70	1	0.6	39	557
S02	JKL Company	0.96	3	2	1.9	35	404
S03	XYZ Company	0.94	52	2	0	26	199
S04	XYZ Company	0.74	17	3	2.2	75	2541
S05	QRS Company	0.48	19	5	0	9	75
S06	QRS Company	0.68	0	5	6.1	92	4132
S07	ABC Company	0.95	60	2	1.5	94	3266
S08	JKL Company	0.39	90	3	0	20	289
S09	ABC Company	0.47	25	1	0	5	6
S10	ABC Company	0.53	.	4	1.6	62	1437

Based on contracts with its suppliers, the retailer must follow an (s, nQ) policy for items S02, S05, and S08. Items S02 and S08 have a fixed ordering cost for each lot ordered, and item S05 has a single fixed ordering cost, independent of the number of lots ordered. In addition, item S06 has a fixed cost of 0, so the retailer follows a base-stock policy. For the remaining items, the retailer follows (s, S) policies.

The retailer faces additional constraints on some items. When placing an order for item S01, the retailer must take into account that the supplier does not fill any orders that are smaller than 15 items. The supplier for item S07 fills orders only in multiples of 10 items. The fixed cost of item S10 is unknown, so the retailer imposes a maximum order frequency of 25% (that is, on average, the retailer orders at most once every four review periods). The retailer also imposes a maximum order frequency of 50% for S04, even though the fixed ordering cost is known for this item.

Given this information, the retailer first wants to calculate inventory policies that have a target fill rate of 97%, which means that 97% of all incoming customer orders can be filled from on-hand inventory. The information is stored in the following data set in1_fr:

```
data in1_fr;
   format sku $3. supplier $11. policyType $2. serviceType $2.;
   input  sku $ supplier & holdingCost fixedCost LTmean LTvar
          RTDmean RTDvar serviceLevel serviceType $
          policyType $ fixedLotSize minOrderSize maxFreq ;
   datalines;
S01 ABC Company  0.78  70  1  0.6  39   557  0.97  FR  SS   .  15     .
S02 JKL Company  0.96   3  2  1.9  35   404  0.97  FR  NQ   .   .     .
S03 XYZ Company  0.94  52  2  0    26   199  0.97  FR  SS   .   .     .
S04 XYZ Company  0.74  17  3  2.2  75  2541  0.97  FR  SS   .   .  0.50
S05 QRS Company  0.48  19  5  0     9    75  0.97  FR  RQ   .   .     .
S06 QRS Company  0.68   0  5  6.1  92  4132  0.97  FR  BS   .   .     .
S07 ABC Company  0.95  60  2  1.5  94  3266  0.97  FR  SS  10   .     .
S08 JKL Company  0.39  90  3  0    20   289  0.97  FR  NQ   .   .     .
S09 ABC Company  0.47  25  1  0     5     6  0.97  FR  SS   .   .     .
S10 ABC Company  0.53   .  4  1.6  62  1437  0.97  FR  SS   .   .  0.25
;
```

The retailer then uses the following call to PROC IRP to compute the inventory policies. Because METHOD=SERVICE, the heuristics that are used to compute inventory policies are based on target service levels. The variables in the input data set are specified in the HOLDINGCOST, ITEMID, LEADTIME, POLICYTYPE, REPLENISHMENT, REVIEWTIMEDEMAND, and SERVICE statements. Note that the ITEMID statement specifies two variables, sku and supplier. The specified variables are copied from the input data set to the output data set; this enables the retailer to include additional information about the suppliers in the output data set.

```
proc irp data=in1_fr out=out1_fr method=service;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   policytype policyType;
   replenishment / fcost=fixedCost lotsize=fixedLotSize
                   minsize=minOrderSize maxfreq=maxFreq;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set is shown in Output 2.1.1. This data set contains two variables that define the computed policy: reorderLevel and orderUpToLevel. The remaining variables give more details about the policy, including statistics regarding average inventory, average back orders, and so on, in addition to the type of algorithm used to compute the policy.

The first two characters in the _ALGORITHM_ variable are FR for all observations because the algorithm used the fill rate target level in the heuristic. The second set of characters in the _ALGORITHM_ variable gives the type of policy computed. The third set of characters in the _ALGORITHM_ variable indicates which distribution is used to approximate both the lead-time demand and the demand during lead time plus review time. This is either GA for the gamma distribution or NO for the normal distribution.

Output 2.1.1 Inventory Policies with 97% Target Fill Rate

PROC IRP Results
Target Measure: 97% Fill Rate

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	119	209	101.694	1.58259	0.38244	106.092
2	S02	JKL Company	203	238	117.655	2.15474	1.00000	115.949
3	S03	XYZ Company	90	147	50.264	0.70077	0.40027	68.062
4	S04	XYZ Company	532	643	315.581	4.26971	0.49855	242.006
5	S05	QRS Company	77	118	43.841	0.34130	0.21771	25.180
6	S06	QRS Company	1121	1122	577.537	7.53671	0.99995	392.725
7	S07	ABC Company	570	580	302.990	5.02517	0.99491	347.535
8	S08	JKL Company	103	228	86.188	0.68808	0.16000	48.013
9	S09	ABC Company	9	32	12.855	0.10577	0.24150	12.079
10	S10	ABC Company	395	662	249.004	2.27842	0.24993	131.972

Obs	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	2.60754	0.040579	0.38350	0.97200	0.95418	FR-SS-GA	SUCCESSFUL
2	3.36156	0.061564	0.29748	0.97005	0.95156	FR-NQ-GA	SUCCESSFUL
3	1.93323	0.026953	0.51727	0.97356	0.94380	FR-SS-NO	SUCCESSFUL
4	4.20775	0.056929	0.23766	0.97033	0.95958	FR-SS-GA	SUCCESSFUL
5	4.87126	0.037922	0.20529	0.96986	0.96356	FR-RQ-NO	SUCCESSFUL
6	6.27757	0.081921	0.15930	0.97062	0.96101	FR-BS-GA	SUCCESSFUL
7	3.22330	0.053459	0.31024	0.97029	0.95450	FR-SS-GA	SUCCESSFUL
8	4.30940	0.034404	0.23205	0.96998	0.95956	FR-NQ-NO	SUCCESSFUL
9	2.57091	0.021154	0.38897	0.97892	0.94960	FR-SS-NO	SUCCESSFUL
10	4.01619	0.036749	0.24899	0.97100	0.95821	FR-SS-NO	SUCCESSFUL

The fill rates for all items are near 97%, the specified target level. However, suppose the retailer thinks that the resulting back-order ratios are unacceptably high. Only one service measure per observation can be specified in a single call to PROC IRP. So now the retailer specifies a 3% target back-order ratio for all items, which ignores the 97% target fill rate. The following DATA step makes this change:

```
data in1_br;
   set in1_fr;
   serviceLevel = 0.03;
   serviceType = 'BR';
run;
```

The retailer then calls PROC IRP as follows. Note that this call is the same as the previous call to PROC IRP except for a different name for the output data set. Some of the variable values (for the serviceLevel and serviceType variables) have changed, but the variable names have not changed.

```
proc irp data=in1_br out=out1_br method=service;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   policytype policyType;
   replenishment / fcost=fixedCost lotsize=fixedLotSize
                   minsize=minOrderSize  maxfreq=maxFreq;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set is shown in Output 2.1.2. Notice that the average inventory increased for the 3% back-order ratio target level, as compared to the 97% fill rate target level. More inventory is required for meeting this more restrictive target service measure.

Output 2.1.2 Inventory Policies with 3% Target Back-Order Ratio

PROC IRP Results
Target Measure: 3% Back-Order Ratio

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	130	220	112.259	1.14802	0.38244	114.333
2	S02	JKL Company	234	269	147.562	1.06201	1.00000	144.660
3	S03	XYZ Company	90	147	50.264	0.70077	0.40027	68.062
4	S04	XYZ Company	600	711	381.532	2.21989	0.49855	290.809
5	S05	QRS Company	79	120	45.775	0.27482	0.21771	26.108
6	S06	QRS Company	1311	1312	762.757	2.75746	0.99995	518.675
7	S07	ABC Company	634	644	364.758	2.79347	0.99491	406.215
8	S08	JKL Company	105	230	88.111	0.61110	0.16000	48.763
9	S09	ABC Company	9	32	12.855	0.10577	0.24150	12.079
10	S10	ABC Company	407	674	260.547	1.82169	0.24993	138.090

Obs	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	2.87845	0.029437	0.34741	0.97997	0.96624	BR-SS-GA	SUCCESSFUL
2	4.21606	0.030343	0.23719	0.98637	0.97542	BR-NQ-GA	SUCCESSFUL
3	1.93323	0.026953	0.51727	0.97356	0.94380	BR-SS-NO	SUCCESSFUL
4	5.08709	0.029599	0.19658	0.98481	0.97833	BR-SS-GA	SUCCESSFUL
5	5.08609	0.030536	0.19661	0.97534	0.96980	BR-RQ-NO	SUCCESSFUL
6	8.29084	0.029972	0.12062	0.98986	0.98510	BR-BS-GA	SUCCESSFUL
7	3.88041	0.029718	0.25771	0.98404	0.97405	BR-SS-GA	SUCCESSFUL
8	4.40556	0.030555	0.22699	0.97313	0.96341	BR-NQ-NO	SUCCESSFUL
9	2.57091	0.021154	0.38897	0.97892	0.94960	BR-SS-NO	SUCCESSFUL
10	4.20237	0.029382	0.23796	0.97644	0.96549	BR-SS-NO	SUCCESSFUL

As the average inventory increases, the average cost also increases because the retailer has not specified a penalty cost for the back orders in order to balance the holding cost of inventory. The only costs that are considered in the service-level heuristics are fixed ordering costs and inventory holding costs. Output 2.1.1 and Output 2.1.2 show that the average ordering frequency (avgOrderFreq) does not change much between the two target-level specifications. Therefore, the bulk of the increase in average cost comes from the increase in average inventory. The retailer now has two policies to choose from. Although one policy does have a higher average cost, the decision should not be based on cost alone. Both policies are heuristic policies and are derived using different target service levels. With a higher level of service comes a higher cost, and the retailer must decide, based on the desired levels of service, which policy best fits the needs of the company.

Another option for the retailer is to use back-order penalty cost information to find inventory policies. This problem is explored in Example 2.2 and Example 2.3.

Example 2.2: Single-Location System: Penalty Costs

In this example, assume that the retailer from Example 2.1 is able to obtain estimates of back-order penalty costs. Rather than using a service-level heuristic, as in Example 2.1, the retailer uses the penalty costs to calculate inventory policies. First, the retailer uses a heuristic method to calculate nearly optimal inventory policies. In Example 2.3, the optimal inventory policy is calculated.

The back-order penalty costs are contained in the following data set:

```
data pcosts;
   format sku $3. penaltyCost;
   input sku $ penaltyCost;
   datalines;
S01   7.4
S02  10.2
S03   8.1
S04   6.6
S05   9.2
S06   9.0
S07   7.1
S08   3.7
S09   5.2
S10  10.8
;
```

This data set is merged with in1_fr to produce the input data set in2. The variables serviceType and serviceLevel are dropped from the in1_fr data set, because they are not needed when penalty costs are used. However, if these variables are left in the data set, they are simply ignored when METHOD=PENALTY.

```
data in2;
   merge in1_fr (drop=serviceType serviceLevel)
         pcosts;
   by sku;
run;
```

Then, the retailer calls PROC IRP by using the following statements. There are several differences between this call and the calls to PROC IRP in Example 2.1. First, METHOD=PENALTY. Second, the PENALTY statement is included, and the penalty cost variable is identified as penaltyCost. There are no other options specified in the PENALTY statement, so the policy is calculated using a heuristic. Finally, the SERVICE statement is no longer listed in the PROC IRP call; if it had been listed, it would be ignored.

```
proc irp data=in2 out=out2 method=penalty;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   penalty / cost=penaltyCost;
   policytype policyType;
   replenishment / fcost=fixedCost lotsize=fixedLotSize
                   minsize=minOrderSize maxfreq=maxFreq;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set is shown in Output 2.2.1. For all items in this example, the average inventory is lower than that shown in Output 2.1.1 and Output 2.1.2, and the average number of back orders is higher. As a result, the fill rates for this policy are less than 97%, and the back-order ratios are greater than 3%.

The average cost of the penalty-cost policy might be higher than that of the previous policies (as is the case for items S03, S05, S09, and S10), lower than that of the previous policies (as is the case for items S04 and S07), or between the average costs of the two previous policies (as is the case for the remaining items). For example, the average cost might be lower because a lower service level is implied by the specified penalty costs. On the other hand, the penalty-cost heuristics include penalty costs for back orders. These costs are not included in the service-level heuristics, so an increase in average cost for the penalty-cost method might result from including this extra cost parameter. Therefore, use caution when comparing output from the service-level method and penalty-cost method, because the two methods use different levels of information to compute costs and determine policies.

Output 2.2.1 Inventory Policies with Penalty-Cost Heuristic

PROC IRP Results
Penalty-Cost Heuristic

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	93	183	77.418	3.3069	0.38244	111.628
2	S02	JKL Company	176	211	92.418	3.9176	1.00000	131.680
3	S03	XYZ Company	81	138	41.947	1.3832	0.40027	71.447
4	S04	XYZ Company	427	538	217.530	11.2186	0.49855	243.491
5	S05	QRS Company	74	115	40.967	0.4669	0.21771	28.096
6	S06	QRS Company	998	999	461.089	14.0894	0.99995	440.345
7	S07	ABC Company	457	467	198.559	13.5939	0.99491	344.842
8	S08	JKL Company	83	208	67.504	2.0040	0.16000	48.141
9	S09	ABC Company	9	32	12.855	0.1058	0.24150	12.629
10	S10	ABC Company	389	656	243.266	2.5412	0.24993	156.376

Obs	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	1.98508	0.08479	0.50376	0.94017	0.90844	PC-SS-GA	SUCCESSFUL
2	2.64050	0.11193	0.37872	0.94287	0.91467	PC-NQ-GA	SUCCESSFUL
3	1.61333	0.05320	0.61984	0.94865	0.90175	PC-SS-NO	SUCCESSFUL
4	2.90041	0.14958	0.34478	0.92173	0.90005	PC-SS-GA	SUCCESSFUL
5	4.55188	0.05188	0.21969	0.95979	0.95229	PC-RQ-NO	SUCCESSFUL
6	5.01184	0.15315	0.19953	0.94397	0.92963	PC-BS-GA	SUCCESSFUL
7	2.11232	0.14462	0.47341	0.91689	0.88398	PC-SS-GA	SUCCESSFUL
8	3.37520	0.10020	0.29628	0.92066	0.90307	PC-NQ-NO	SUCCESSFUL
9	2.57091	0.02115	0.38897	0.97892	0.94960	PC-SS-NO	SUCCESSFUL
10	3.92365	0.04099	0.25486	0.96791	0.95415	PC-SS-NO	SUCCESSFUL

Example 2.3: Single-Location System: OPTIMAL Option

In Example 2.2, the retailer uses penalty costs to compute nearly optimal inventory replenishment policies. By specifying the OPTIMAL option in the PENALTY statement, you can use back-order penalty costs to compute optimal policies. PROC IRP computes the optimal reorder level and order-up-to-level within the class of policy specified by the policyType variable (that is, SS, BS, NQ, or RQ).

The call to PROC IRP is shown in the following statements. The OPTIMAL option is specified in the PENALTY statement. In addition, the LOTSIZE=, MINSIZE=, and MAXFREQ= options are no longer included in the REPLENISHMENT statement, because these options are ignored when the OPTIMAL option is used.

```
proc irp data=in2 out=out3 method=penalty;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   penalty / cost=penaltyCost optimal;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set is shown in Output 2.3.1. Notice that the average cost for most items (except for S03, S05, and S08) is lower than the average cost in Output 2.2.1. This is expected, because the OPTIMAL option finds the optimal (that is, lowest-cost) inventory replenishment policy. However, the average cost for the remaining items actually rises. There are two reasons why this might happen. First, the penalty-cost heuristic given in Output 2.2.1 uses an approximation of the cost of the policy; the actual cost might be slightly higher or lower than the value given in the avgCost variable. Moreover, the heuristic uses either a gamma distribution or a normal distribution to approximate both the lead-time demand and the demand during lead time plus review time, whereas the optimization uses either a negative binomial distribution or a shifted Poisson distribution. Therefore, the underlying assumptions of the models are different, and care should be used in comparing results across the two models. The policy that is calculated using the OPTIMAL option is the optimal policy with respect to the lead time and demand distributions used by PROC IRP, but it might reflect a higher cost than a policy that is calculated using different distributions for lead time and demand.

In this example, the negative binomial distribution is used for the demand during lead time plus review time of all items, as indicated by a B in the fifth character of the _ALGORITHM_ variable. This distribution is also used for the review-time demand, as indicated by a B in the sixth character of the _ALGORITHM_ variable. Note that the sixth character of the _ALGORITHM_ variable is '_' for items that follow an NQ policy. This indicates that the review-time demand distribution does not play a role in the optimization algorithm.

Recall from Example 2.1 that the fixed cost for item S10 was not easily estimated, so a maximum ordering frequency was used instead. However, the OPTIMAL option ignores the LOTSIZE=, MINSIZE=, and MAXFREQ= options, so item S10 is no longer constrained by a maximum ordering frequency of 25%. In addition, because the fixed cost for item S10 was not specified, PROC IRP assumes that it is zero. As a result, the policy for S10 in Output 2.3.1 is a base-stock policy (as indicated by the BS in the _ALGORITHM_ variable), and the reorderLevel and orderUpToLevel values are quite different from those in the previous examples. However, the original intention of including a missing value for the fixed cost for S10 was to account for the fact that the cost was unknown rather than to imply that the cost was zero. Therefore, when using the OPTIMAL option, you should specify estimates for fixed costs of all items, unless the fixed cost is assumed to be zero.

Output 2.3.1 Inventory Policies with the OPTIMAL Option

PROC IRP Results
Penalty Cost with OPTIMAL Option

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	93	188	77.538	3.4709	0.32194	108.700
2	S02	JKL Company	173	213	92.623	3.8979	0.87719	131.308
3	S03	XYZ Company	83	138	41.674	1.7570	0.36451	72.360
4	S04	XYZ Company	441	525	214.188	11.2445	0.54722	242.016
5	S05	QRS Company	76	117	43.684	0.6844	0.21791	31.405
6	S06	QRS Company	999	1000	461.055	13.9348	0.99252	438.931
7	S07	ABC Company	375	522	206.377	13.8543	0.44856	321.338
8	S08	JKL Company	83	208	68.445	2.4453	0.16000	50.141
9	S09	ABC Company	8	31	11.429	0.2308	0.19531	11.455
10	S10	ABC Company	526	527	220.297	3.2970	0.99800	152.365

Obs	inventory Ratio	backorder Ratio	turnover	ready Rate	_scale_	_algorithm_	_status_
1	1.98815	0.08900	0.50298	0.90281	1.00	PC-SS-BB	SUCCESSFUL
2	2.64637	0.11137	0.37788	0.91369	1.05	PC-NQ-B_	SUCCESSFUL
3	1.60286	0.06758	0.62388	0.89413	1.00	PC-SS-BB	SUCCESSFUL
4	2.85584	0.14993	0.35016	0.89768	3.00	PC-SS-BB	SUCCESSFUL
5	4.85382	0.07605	0.20602	0.94879	1.00	PC-RQ-BB	SUCCESSFUL
6	5.01147	0.15147	0.19954	0.92892	5.52	PC-BS-BB	SUCCESSFUL
7	2.19550	0.14739	0.45548	0.88118	2.82	PC-SS-BB	SUCCESSFUL
8	3.42226	0.12226	0.29220	0.90366	1.00	PC-NQ-B_	SUCCESSFUL
9	2.28584	0.04615	0.43748	0.90048	1.00	PC-SS-BB	SUCCESSFUL
10	3.55318	0.05318	0.28144	0.95311	3.10	PC-BS-BB	SUCCESSFUL

Example 2.4: Single-Location System: LEADTIMEDEMAND Statement

This example illustrates the use of PROC IRP for a retailer who faces a nonstationary demand with a lead time that is longer than the review period. The IRP procedure uses the review-time demand and lead-time information to calculate the parameters of lead-time demand. When demand is nonstationary (that is, demand fluctuates over time), it is not sufficient to know just the lead time and mean review-time demand information. In such situations, you can directly specify the mean and variance of lead-time demand by using the LEADTIMEDEMAND statement.

For example, suppose the lead time for an item is three periods, but the demands over the next four review periods are 25, 32, 40, and 28. If the mean of the review-time demand is specified as 25 (the mean of the current period's demand), and the lead-time mean is specified as 3 in the LEADTIME statement, PROC IRP computes the mean lead-time demand as 75 (= 3×25). This is an inaccurate calculation of lead-time demand, because it does not account for the fluctuations in demand in the subsequent periods. Rather, the correct calculation of lead-time demand is the demand over the next three periods after the current review period, which is 100 (the sum of 32, 40, and 28). This example illustrates how the LEADTIMEDEMAND statement overcomes such a problem.

In the following DATA step, the data set in4 gives the input to PROC IRP. The mean and variance of lead-time demand are given by the LTDmean and LTDvar variables, respectively. The mean and variance of review-time demand are given by the RTDmean and RTDvar variables, respectively. Note that the mean and variance of lead time are not included in this data set. When the LEADTIMEDEMAND statement is used, these variables are not used.

```
data in4;
   format sku $3.;
   input  sku $ holdingCost fixedCost
          LTDmean LTDvar RTDmean RTDvar
          serviceLevel;
datalines;
B01  0.52  62  100   894  25   56  0.95
B02  0.86  17   80   633  50  227  0.95
B03  0.27  48  275  4101  90  506  0.95
B04  0.94  23   64   719  15   38  0.95
B05  0.62  38   90  1188  32  163  0.95
B06  0.44  82  122  4324  52  675  0.95
B07  0.75  68  170  2823  84  632  0.95
B08  0.78  73   30   365  10   35  0.95
B09  0.46  18   91   989  66  533  0.95
B10  0.55  25  144  3741  71  807  0.95
;
```

The following call to PROC IRP computes inventory replenishment policies by using a 95% target fill rate. In the PROC IRP statement, the METHOD= option is not specified, so the default value of SERVICE is used. The HOLDINGCOST statement is not required because the holding cost variable is named holdingCost, the default name for PROC IRP. Because the POLICYTYPE statement is not specified, PROC IRP computes (s, S) policies for all items in the data set. In addition, the REPLENISHMENT statement is not required because the fixed cost variable is named fixedCost, the default name for PROC IRP. Finally, the SERVICE statement is not specified, so PROC IRP uses the fill rate (the default service measure) for all items.

```
proc irp data=in4 out=out4;
   itemid sku;
   leadtimedemand / mean=LTDmean variance=LTDvar;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set is shown in Output 2.4.1. Because the LEADTIMEDEMAND statement was used, these policies should be interpreted as the policies to follow only for the current review period. Because demand is nonstationary and the lead times are longer than the review period, you should compute new policies each period, using updated information about lead-time demand and review-time demand.

Output 2.4.1 Inventory Policies with the LEADTIMEDEMAND Statement

PROC IRP Results
Target Measure: 95% Fill Rate

Obs	sku	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost	inventory Ratio
1	B01	131	214	58.397	1.41382	0.30916	49.535	2.33588
2	B02	130	177	40.256	2.35698	0.75140	47.394	0.80511
3	B03	356	539	118.050	4.52323	0.44883	53.417	1.31166
4	B04	102	134	44.889	0.88307	0.41680	51.782	2.99259
5	B05	137	205	61.300	1.65391	0.41876	53.919	1.91563
6	B06	209	361	134.347	3.64541	0.33104	86.258	2.58360
7	B07	257	385	94.237	4.10412	0.52055	106.075	1.12186
8	B08	51	101	41.743	0.93225	0.22602	49.059	4.17433
9	B09	156	231	57.522	3.22583	0.63384	37.869	0.87154
10	B10	247	334	98.831	3.57699	0.59779	69.302	1.39199

Obs	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	0.056553	0.42810	0.95271	0.91768	FR-SS-NO	SUCCESSFUL
2	0.047140	1.24207	0.95327	0.86847	FR-SS-NO	SUCCESSFUL
3	0.050258	0.76239	0.95215	0.89042	FR-SS-NO	SUCCESSFUL
4	0.058872	0.33416	0.95552	0.93183	FR-SS-NO	SUCCESSFUL
5	0.051685	0.52202	0.95373	0.91486	FR-SS-NO	SUCCESSFUL
6	0.070104	0.38706	0.95191	0.92702	FR-SS-GA	SUCCESSFUL
7	0.048859	0.89137	0.95231	0.88285	FR-SS-NO	SUCCESSFUL
8	0.093225	0.23956	0.95425	0.93756	FR-SS-GA	SUCCESSFUL
9	0.048876	1.14740	0.95139	0.86808	FR-SS-NO	SUCCESSFUL
10	0.050380	0.71840	0.95214	0.89932	FR-SS-NO	SUCCESSFUL

Example 2.5: Continuous Review Approximation: Review Period Shorter Than Forecast Interval

This example considers a retailer who forecasts demand data on a monthly basis but reviews inventory on a weekly basis. For the purpose of this illustration, it is assumed that there are exactly four weeks in a month.

For example, consider Table 2.1 in the section "Getting Started" on page 6, but suppose that the mean and variance of demand specify the demand over one month. In addition, suppose that the lead time of all items is one week (the same as the review period). This is not an example of continuous review, because the retailer still makes decisions at discrete time periods. However, it might be considered an approximation of a continuous review system, because decisions are made at points throughout the demand forecast interval. If smaller review periods (for example, one day or one hour) are chosen, this becomes a closer approximation of a continuous review system.

The data for this example are given in the following data set, data5. The variables MeanOfDemand and VarianceOfDemand give the mean and variance of demand over an entire month.

```
data data5;
   format Sku $1.;
   input  Sku $ HoldingCost OrderingCost
          LeadTime MeanOfDemand VarianceOfDemand;
datalines;
A  0.35  90  1  125.1  2170.8
B  0.05  50  1  140.3  1667.7
C  0.12  50  1  116.0  3213.4
D  0.10  75  1  291.8  5212.4
E  0.45  75  1  134.5  1980.5
;
```

This data set is transformed to the input data set for PROC IRP by using the following DATA step. From the assumption that there are four weeks in a month, the mean and variance of review-time demand (RTDmean and RTDvar, respectively) are calculated by dividing MeanOfDemand by 4 and VarianceOfDemand by 16. For this calculation to be valid, the demand for one month must be assumed to be uniform over the entire month, so that the demand for a single week is one-quarter of the monthly demand.

```
data in5;
   set data5;
   RTDmean = MeanOfDemand / 4 ;
   RTDvar = VarianceOfDemand / 16 ;
   serviceLevel = 0.96 ;
run;
```

The call to PROC IRP is as follows. Notice that the VARIANCE= option is not specified in the LEADTIME statement, because the lead times are assumed to be deterministic (that is, to have zero variance).

```
proc irp data=in5 out=out5 method=service;
   holdingcost HoldingCost;
   itemid Sku;
   leadtime / mean=LeadTime;
   replenishment / fcost=OrderingCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel;
run;
```

The output data set is shown in Output 2.5.1.

Output 2.5.1 Inventory Policies When Review Period Is Shorter Than Forecast Interval

PROC IRP Results
Target Measure: 96% Fill Rate

Obs	Sku	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost	inventory Ratio
1	A	47	169	59.268	1.10664	0.27373	45.3798	1.89505
2	B	35	277	105.737	1.29462	0.18716	14.6450	3.01461
3	C	42	192	73.833	1.11256	0.22262	19.9912	2.54598
4	D	95	404	135.546	2.82158	0.25877	32.9626	1.85807
5	E	53	155	50.198	1.19067	0.32802	47.1905	1.49287

Obs	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	0.035384	0.52769	0.96471	0.91056	FR-SS-NO	SUCCESSFUL
2	0.036910	0.33172	0.96317	0.91166	FR-SS-NO	SUCCESSFUL
3	0.038364	0.39278	0.96206	0.92196	FR-SS-NO	SUCCESSFUL
4	0.038678	0.53819	0.96133	0.88895	FR-SS-NO	SUCCESSFUL
5	0.035410	0.66985	0.96463	0.89963	FR-SS-NO	SUCCESSFUL

Example 2.6: Two-Echelon System: Service-Level Heuristic

This example illustrates the use of PROC IRP and service-level heuristics to compute inventory replenishment policies for a two-echelon-distribution inventory system. Example 2.7 then explores the same two-echelon system with penalty costs.

This example has two warehouses and four retailers. The two items (M01 and M02) can be classified further by color (blue or red). These items could be identified by four distinct values of the SKU variable; however, to illustrate the use of the ITEMID statement, they are identified by SKU (M01 or M02) and color (blue or red). Warehouse W01 supplies item M01, and warehouse W02 supplies item M02. From warehouse W01, only retailers R01 and R02 require blue items, whereas retailers R01, R03, and R04 require red items. From warehouse W02, all four retailers (R01, R02, R03, and R04) require blue items, whereas only retailer R03 requires red items.

Table 2.10 shows the estimates of the holding and fixed costs, mean and variance of the lead time, and mean and variance of the review-time demand. Observations that have a missing value for Retailer correspond to warehouses. For example, the first observation gives the holding cost and fixed cost at warehouse W01, in addition to the mean and variance of lead time from an external supplier to this warehouse. The mean and variance of review-time demand are missing for all the warehouse observations, because the warehouses do not see any external demand apart from the orders placed by the retailers.

The remaining observations correspond to retailers. The missing values of Fixed Cost indicate that the retailers incur no fixed cost for placing an order; therefore, the retailers follow base-stock policies. For these observations, the mean and variance of lead time give data about the lead time from the warehouse to the retailer. In this problem, the lead time variance between the warehouses and the retailers is assumed to be zero, but positive variances can also be used in the two-echelon system. The review-time demand data give the mean and variance of the demand for that item (SKU and color) at that retailer.

Table 2.10 Data Estimates for Two-Echelon System

SKU	Color	Warehouse	Retailer	Holding Cost	Fixed Cost	Lead Time Mean	Lead Time Variance	Review-Time Demand Mean	Review-Time Demand Variance
M01	Blue	W01	.	0.20	61	1	0.12	.	.
M01	Blue	W01	R01	0.75	.	2	0	67	121
M01	Blue	W01	R02	1.42	.	1	0	23	87
M01	Red	W01	.	0.20	61	1	0.12	.	.
M01	Red	W01	R01	0.75	.	2	0	50	793
M01	Red	W01	R03	1.11	.	3	0	42	109
M01	Red	W01	R04	0.65	.	2	0	91	1267
M02	Blue	W02	.	0.17	88	1	0.41	.	.
M02	Blue	W02	R01	0.70	.	1	0	84	931
M02	Blue	W02	R02	1.35	.	2	0	59	1018
M02	Blue	W02	R03	1.04	.	1	0	71	775
M02	Blue	W02	R04	0.62	.	2	0	113	1689
M02	Red	W02	.	0.17	88	1	0.41	.	.
M02	Red	W02	R03	1.04	.	1	0	85	1954

Based on this information, inventory policies are calculated using (s, S) policies for the warehouses and a target fill rate of 97% for the retailers. This means that 97% of all incoming customer orders (to the retailers) can be filled from on-hand inventory. The information is stored in the following data set in6_fr:

```
data in6_fr;
   format warehouse $3. retailer $3. sku $3. color $4.
          policyType $2. serviceType $2. ;
   input sku $ color $ warehouse $ retailer $
         holdingCost fixedCost
         LTmean LTvar RTDmean RTDvar
         policyType $ serviceType $ serviceLevel;
   datalines;
M01  BLUE  W01    .  0.20  61  1  0.12     .     .  SS   .     .
M01  BLUE  W01  R01  0.75   .  2  0       67   121   .  FR  0.97
M01  BLUE  W01  R02  1.42   .  1  0       23    87   .  FR  0.97
M01  RED   W01    .  0.20  61  1  0.12     .     .  SS   .     .
M01  RED   W01  R01  0.75   .  2  0       50   793   .  FR  0.97
M01  RED   W01  R03  1.11   .  3  0       42   109   .  FR  0.97
M01  RED   W01  R04  0.65   .  2  0       91  1267   .  FR  0.97
M02  BLUE  W02    .  0.17  88  1  0.41     .     .  SS   .     .
M02  BLUE  W02  R01  0.70   .  1  0       84   931   .  FR  0.97
M02  BLUE  W02  R02  1.35   .  2  0       59  1018   .  FR  0.97
M02  BLUE  W02  R03  1.04   .  1  0       71   775   .  FR  0.97
M02  BLUE  W02  R04  0.62   .  2  0      113  1689   .  FR  0.97
M02  RED   W02    .  0.17  88  1  0.41     .     .  SS   .     .
M02  RED   W02  R03  1.04   .  1  0       85  1954   .  FR  0.97
;
```

The following call to PROC IRP computes the inventory policies. Because METHOD=SERVICE, heuristics are used to compute inventory policies based on target service levels (in this case, 97% fill rate). The ITEMID statement specifies sku, color, and warehouse as the variables by which to group the items. The LOCATION statement specifies the retailer variable. The remaining variables in the input data set are specified using the HOLDINGCOST, LEADTIME, POLICYTYPE, REPLENISHMENT, REVIEWTIMEDEMAND, and SERVICE statements.

```
proc irp data=in6_fr out=out6_fr method=service;
   holdingcost holdingCost;
   itemid sku color warehouse;
   leadtime / mean=LTmean variance=LTvar;
   location retailer;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set is shown in Output 2.6.1. This data set contains two variables that define the computed policy: reorderLevel and orderUpToLevel. The retailers follow base-stock policies, so the orderUpToLevel is one more than the reorderLevel for the retailers. Note that the value of fillRate for the retailers is very near 97%, the target service level. The fill rate for the warehouses is lower, but the fill rate at the retailers is the main concern because customers are seen only at the retailers.

Output 2.6.1 Inventory Policies with 97% Target Fill Rate

PROC IRP Results for Two-Echelon System
Target Measure: 97% Fill Rate

Obs	sku	color	warehouse	retailer	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq
1	M01	BLUE	W01		138	365	101.849	5.4317	0.36590
2	M01	BLUE	W01	R01	227	228	25.029	2.0728	1.00000
3	M01	BLUE	W01	R02	67	68	21.321	0.7094	1.00000
4	M01	RED	W01		260	593	137.053	17.1120	0.47980
5	M01	RED	W01	R01	245	246	92.928	1.6038	1.00000
6	M01	RED	W01	R03	202	203	32.322	1.2494	1.00000
7	M01	RED	W01	R04	382	383	104.305	2.8142	1.00000
8	M02	BLUE	W02		588	1182	363.979	22.1052	0.48857
9	M02	BLUE	W02	R01	242	243	71.868	2.5462	1.00000
10	M02	BLUE	W02	R02	282	283	103.859	1.8473	1.00000
11	M02	BLUE	W02	R03	210	211	66.396	2.1958	1.00000
12	M02	BLUE	W02	R04	460	461	117.812	3.4509	1.00000
13	M02	RED	W02		160	463	182.018	5.6619	0.28765
14	M02	RED	W02	R03	291	292	119.059	2.7210	1.00000

Obs	avgCost	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	91.738	1.13165	0.060352	0.88366	0.94121	0.84200	__-SS-GA	SUCCESSFUL
2	18.772	0.37357	0.030937	2.67688	0.96907	0.84331	FR-BS-GA	SUCCESSFUL
3	30.276	0.92701	0.030845	1.07873	0.96950	0.91825	FR-BS-GA	SUCCESSFUL
4	230.050	0.74892	0.093508	1.33525	0.91030	0.78621	__-SS-GA	SUCCESSFUL
5	69.696	1.85857	0.032075	0.53805	0.97041	0.94846	FR-BS-GA	SUCCESSFUL
6	35.878	0.76957	0.029748	1.29942	0.97049	0.90239	FR-BS-GA	SUCCESSFUL
7	67.798	1.14621	0.030925	0.87244	0.96991	0.92666	FR-BS-GA	SUCCESSFUL
8	437.483	1.11309	0.067600	0.89840	0.94393	0.87098	__-SS-GA	SUCCESSFUL
9	50.307	0.85557	0.030311	1.16881	0.97006	0.91412	FR-BS-GA	SUCCESSFUL
10	140.210	1.76032	0.031310	0.56808	0.97084	0.94707	FR-BS-GA	SUCCESSFUL
11	69.052	0.93516	0.030927	1.06934	0.96956	0.91862	FR-BS-GA	SUCCESSFUL
12	73.044	1.04259	0.030539	0.95915	0.97010	0.92201	FR-BS-GA	SUCCESSFUL
13	180.077	2.14139	0.066611	0.46699	0.94745	0.91374	__-SS-GA	SUCCESSFUL
14	123.821	1.40069	0.032012	0.71393	0.96954	0.93921	FR-BS-GA	SUCCESSFUL

Suppose that the target service measure for retailer R01 is instead specified as a 3% back-order ratio. The remaining retailers continue to follow policies based on a 97% target fill rate. The DATA step to change the serviceType and serviceLevel variables is as follows:

```
data in6_br;
   set in6_fr;
   if retailer = 'R01' then do;
      serviceType = 'BR';
      serviceLevel = 0.03;
   end;
run;
```

The following call to PROC IRP is exactly the same as the previous call to PROC IRP, except for a different name for the output data set. Some of the variable values (for the serviceLevel and serviceType variables) have changed, but the variable names have not changed.

```
proc irp data=in6_br out=out6_br method=service;
   holdingcost holdingCost;
   itemid sku color warehouse;
   leadtime / mean=LTmean variance=LTvar;
   location retailer;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set is shown in Output 2.6.2. The values of the backorderRatio variable are close to 3% for the three observations that correspond to retailer R01. Note that when the policies for retailer R01 change, the policies for the other retailers and the policies for the warehouses might also change as a result of the changes in the target service level for R01.

Output 2.6.2 Inventory Policies with 3% Target Back-Order Ratio for R01

PROC IRP Results for Two-Echelon System
Target Measure: 3% Back-Order Ratio

Obs	sku	color	warehouse	retailer	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq
1	M01	BLUE	W01		142	369	105.242	4.8250	0.36590
2	M01	BLUE	W01	R01	227	228	25.235	1.9366	1.00000
3	M01	BLUE	W01	R02	67	68	21.416	0.6863	1.00000
4	M01	RED	W01		271	604	145.821	14.8801	0.47980
5	M01	RED	W01	R01	246	247	94.420	1.4857	1.00000
6	M01	RED	W01	R03	201	202	31.817	1.2321	1.00000
7	M01	RED	W01	R04	379	380	102.451	2.8499	1.00000
8	M02	BLUE	W02		588	1182	363.979	22.1052	0.48857
9	M02	BLUE	W02	R01	242	243	71.868	2.5462	1.00000
10	M02	BLUE	W02	R02	282	283	103.859	1.8473	1.00000
11	M02	BLUE	W02	R03	210	211	66.396	2.1958	1.00000
12	M02	BLUE	W02	R04	460	461	117.812	3.4509	1.00000
13	M02	RED	W02		160	463	182.018	5.6619	0.28765
14	M02	RED	W02	R03	291	292	119.059	2.7210	1.00000

Obs	avgCost	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	92.705	1.16936	0.053611	0.85517	0.94768	0.85460	__-SS-GA	SUCCESSFUL
2	18.926	0.37665	0.028904	2.65502	0.97110	0.84992	BR-BS-GA	SUCCESSFUL
3	30.410	0.93111	0.029837	1.07398	0.97047	0.92021	FR-BS-GA	SUCCESSFUL
4	231.157	0.79684	0.081312	1.25496	0.92177	0.80780	__-SS-GA	SUCCESSFUL
5	70.815	1.88840	0.029714	0.52955	0.97247	0.95168	BR-BS-GA	SUCCESSFUL
6	35.317	0.75755	0.029335	1.32005	0.97087	0.90220	FR-BS-GA	SUCCESSFUL
7	66.593	1.12583	0.031318	0.88823	0.96948	0.92511	FR-BS-GA	SUCCESSFUL
8	437.483	1.11309	0.067600	0.89840	0.94393	0.87098	__-SS-GA	SUCCESSFUL
9	50.307	0.85557	0.030311	1.16881	0.97006	0.91412	BR-BS-GA	SUCCESSFUL
10	140.210	1.76032	0.031310	0.56808	0.97084	0.94707	FR-BS-GA	SUCCESSFUL
11	69.052	0.93516	0.030927	1.06934	0.96956	0.91862	FR-BS-GA	SUCCESSFUL
12	73.044	1.04259	0.030539	0.95915	0.97010	0.92201	FR-BS-GA	SUCCESSFUL
13	180.077	2.14139	0.066611	0.46699	0.94745	0.91374	__-SS-GA	SUCCESSFUL
14	123.821	1.40069	0.032012	0.71393	0.96954	0.93921	FR-BS-GA	SUCCESSFUL

Example 2.7: Two-Echelon System: Penalty Costs

This example assumes that estimates of back-order penalty costs are known for the problem in Example 2.6. Rather than using service-level heuristics as in Example 2.6, this example uses a penalty-cost heuristic to calculate inventory policies.

The penalty costs are given in the following data set. There are no penalty costs for back orders at the warehouses, because customers are seen only at the retailers.

```
data pcosts;
   format warehouse $3. retailer $3. sku $3. color $4. ;
   input sku $ color $ warehouse $ retailer $ penaltyCost;
   datalines;
M01  BLUE  W01    .     .
M01  BLUE  W01  R01   6.7
M01  BLUE  W01  R02  10.2
M01  RED   W01    .     .
M01  RED   W01  R01   8.4
M01  RED   W01  R03   5.6
M01  RED   W01  R04   9.1
M02  BLUE  W02    .     .
M02  BLUE  W02  R01   3.4
M02  BLUE  W02  R02   6.9
M02  BLUE  W02  R03   7.7
M02  BLUE  W02  R04  12.4
M02  RED   W02    .     .
M02  RED   W02  R03   7.5
;
```

This data set is merged with in6_fr to produce the input data set in7. The variables serviceLevel and serviceType are dropped from the in6_fr data set, because they are not needed when penalty costs are used. However, if these variables are left in the data set, they are simply ignored when METHOD=PENALTY.

```
data in7;
   merge in6_fr (drop=serviceLevel serviceType)
         pcosts;
   by sku color warehouse retailer;
run;
```

There are two main differences between the following call to PROC IRP and the call in Example 2.6: First, METHOD=PENALTY is specified in the PROC IRP statement to indicate that a penalty-cost heuristic should be used to compute the policies. Also, the SERVICE statement is removed, and the PENALTY statement is added to specify the variable in the input data set that gives the penalty costs.

```
proc irp data=in7 out=out7 method=penalty;
   holdingcost holdingCost;
   itemid sku color warehouse;
   leadtime / mean=LTmean variance=LTvar;
   location retailer;
   penalty / cost=penaltyCost;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set is shown in Output 2.7.1.

Output 2.7.1 Inventory Policies with Penalty-Cost Heuristic

PROC IRP Results for Two-Echelon System
Penalty-Cost Heuristic

Obs	sku	color	warehouse	retailer	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq
1	M01	BLUE	W01		150	377	112.176	3.7594	0.36590
2	M01	BLUE	W01	R01	231	232	29.274	1.1642	1.00000
3	M01	BLUE	W01	R02	62	63	17.135	1.1271	1.00000
4	M01	RED	W01		279	612	152.344	13.4028	0.47980
5	M01	RED	W01	R01	227	228	76.983	2.6447	1.00000
6	M01	RED	W01	R03	192	193	24.233	2.3090	1.00000
7	M01	RED	W01	R04	382	383	105.852	2.5170	1.00000
8	M02	BLUE	W02		614	1208	386.836	18.9625	0.48857
9	M02	BLUE	W02	R01	216	217	49.642	5.5133	1.00000
10	M02	BLUE	W02	R02	234	235	61.109	6.5300	1.00000
11	M02	BLUE	W02	R03	197	198	55.173	3.2907	1.00000
12	M02	BLUE	W02	R04	481	482	138.413	1.9658	1.00000
13	M02	RED	W02		172	475	193.064	4.7078	0.28765
14	M02	RED	W02	R03	254	255	85.936	5.7170	1.00000

Obs	avgCost	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	110.339	1.24641	0.04177	0.80231	0.95911	0.87870	__-SS-GA	SUCCESSFUL
2	29.756	0.43693	0.01738	2.28871	0.98262	0.89901	PC-BS-GA	SUCCESSFUL
3	35.828	0.74500	0.04901	1.34228	0.95155	0.87425	PC-BS-GA	SUCCESSFUL
4	271.227	0.83248	0.07324	1.20123	0.92940	0.82275	__-SS-GA	SUCCESSFUL
5	79.953	1.53966	0.05289	0.64950	0.95157	0.91782	PC-BS-GA	SUCCESSFUL
6	39.829	0.57698	0.05498	1.73317	0.94555	0.83212	PC-BS-GA	SUCCESSFUL
7	91.709	1.16321	0.02766	0.85969	0.97298	0.93254	PC-BS-GA	SUCCESSFUL
8	482.715	1.18299	0.05799	0.84532	0.95195	0.88700	__-SS-GA	SUCCESSFUL
9	53.495	0.59098	0.06563	1.69211	0.93533	0.82819	PC-BS-GA	SUCCESSFUL
10	127.554	1.03574	0.11068	0.96549	0.90022	0.83484	PC-BS-GA	SUCCESSFUL
11	82.718	0.77709	0.04635	1.28685	0.95440	0.88158	PC-BS-GA	SUCCESSFUL
12	110.192	1.22489	0.01740	0.81640	0.98287	0.95239	PC-BS-GA	SUCCESSFUL
13	190.384	2.27134	0.05539	0.44027	0.95623	0.92693	__-SS-GA	SUCCESSFUL
14	132.251	1.01101	0.06726	0.98911	0.93632	0.87851	PC-BS-GA	SUCCESSFUL

References

Ehrhardt, R., and Mosier, C. (1984). "A Revision of the Power Approximation for Computing (s, S) Policies." *Management Science* 30:618–622.

Graves, S. C., Rinnooy Kan, A. H. G., and Zipkin, P. H., eds. (1993). *Logistics of Production and Inventory*. Vol. 4 of Handbooks in Operations Research and Management Science. Amsterdam: North-Holland/Elsevier Science.

Matta, K. F., and Sinha, D. (1995). "Policy and Cost Approximations of Two-Echelon Distribution Systems with a Procurement Cost at the Higher Echelon." *IIE Transactions* 27:638–645.

Schneider, H. (1978). "Methods for Determining the Re-order Point of an (s, S) Ordering Policy When a Service Level Is Specified." *Journal of the Operational Research Society* 29:1181–1193.

Schneider, H. (1981). "Effects of Service-Levels on Order-Points or Order-Levels in Inventory Models." *International Journal of Production Research* 19:615–631.

Schneider, H., and Ringuest, J. L. (1990). "Power Approximation for Computing (s, S) Policies Using Service Level." *Management Science* 36:822–834.

Silver, E. A., Pyke, D. F., and Peterson, R. (1998). *Inventory Management and Production Planning and Scheduling*. New York: John Wiley & Sons.

Svoronos, A., and Zipkin, P. H. (1991). "Evaluation of One-for-One Replenishment Policies for Multiechelon Inventory Systems." *Management Science* 37:68–83.

Tijms, H., and Groenevelt, H. (1984). "Simple Approximations for the Reorder Point in Periodic and Continuous Review (s, S) Inventory Systems with Service Level Constraints." *European Journal of Operational Research* 17:175–190.

Zheng, Y., and Chen, F. (1992). "Inventory Policies with Quantized Ordering." *Naval Research Logistics* 39:285–305.

Zheng, Y., and Federgruen, A. (1992). "Finding Optimal (s, S) Policies Is About as Simple as Evaluating a Single Policy." *Operations Research* 39:654–665.

Zipkin, P. H. (2000). *Foundations of Inventory Management*. New York: McGraw-Hill.

Chapter 3
The MIRP Procedure

Contents

Overview

The MIRP procedure is designed for inventory replenishment planning. It answers three basic questions about inventory:

1. Where should it be stocked?
2. When should it be stocked?
3. How much of it should be stocked?

A replenishment plan consists of control parameters that determine replenishment quantities for each product at each location at each period. The MIRP procedure optimizes these control parameters so that service-level requirements are satisfied at minimum inventory costs.

There are two inventory control theories: discrete-time theory and continuous-time theory. Discrete-time theory assumes that inventory replenishment orders occur only periodically (that is, only at certain time points). Continuous-time theory enables replenishment orders to occur at any time.

Discrete-time theory is closer to practice and is implemented in the MIRP procedure. Because replenishment orders occur periodically, a *base period* must be defined. The base period is the time between two replenishment orders. It could be any time unit, such as one day, one week, or one month. For simplicity, in this documentation the term *period* refers to the base period.

Getting Started

This section discusses four examples that address three different network structures: a single location, a two-echelon distribution network, and a two-echelon assembly network. Each example starts with a problem description. Input data sets are given to demonstrate how each problem can be modeled. Then procedure statements and output data sets are presented. Each of the first three examples solves for only one network. The fourth example demonstrates how to organize input data sets so that a single call to PROC MIRP solves for multiple networks.

Single Location

A product *R* is sold at a store *W*. Replenishment orders are placed once every Monday morning and are received one week later from an external supplier. The product faces a stable demand in the next eight weeks, with a mean of 10 and a variance of 9 per week. Demand is filled with on-hand inventory. Any unsatisfied demand is backlogged, and the cost of any unsold products to be kept at the store is 1 per unit per week. The store owner wants to satisfy customer demand at a fill rate of 95%. This means that 95% of all incoming customer orders can be filled from on-hand inventory.

You can use the MIRP procedure to determine optimal replenishment parameters that meet the service-level requirement at a minimum inventory holding cost. PROC MIRP requires three input data sets, as follows:

```
data nodedata1;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         leadtime servicelevel holdingcost;
datalines;
N1 RW 1 0.95 1
;

data arcdata1;
   format networkid $2. predecessor successor $8.;
   input networkid $3. predecessor $9. successor $8.;
datalines;
N1 EXTERNAL RW
;

data demanddata1;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N1 RW 1 10 9
N1 RW 2 10 9
N1 RW 3 10 9
N1 RW 4 10 9
N1 RW 5 10 9
N1 RW 6 10 9
N1 RW 7 10 9
N1 RW 8 10 9
;
```

Because replenishments occur on a weekly basis, the base period is defined as one week. The nodedata1 data set contains data, such as name, lead time, service-level requirement, and unit holding cost, that are associated with a SKU-location (that is, a node) in a network. The arcdata1 data set describes network structures by defining arcs (linkages between nodes) between predecessors and successors. The demanddata1 data set holds the forecast of customer demand over the next eight weeks.

The NetworkID variable specifies the name of a network that a SKU-location or an arc belongs to. This variable enables you to group inputs of multiple independent networks by issuing a single call to the procedure.

The EXTERNAL keyword in the arcdata1 data set is reserved for external suppliers. If a SKU at a location is purchased from an outside supplier, an arc from EXTERNAL to the location must be included in the data set.

The following call to PROC MIRP computes optimal inventory levels for product *R* at store *W*:

```
proc mirp nodedata=nodedata1 arcdata=arcdata1 demanddata=demanddata1
          out=out_example1 HORIZON=8;
   NODE / NETWORKID=networkid SKULOC=skuloc LEADTIME=leadtime
          SERVICELEVEL=servicelevel HOLDINGCOST=holdingcost;
    ARC / NETWORKID=networkid PREDECESSOR=predecessor SUCCESSOR=successor;
 DEMAND / NETWORKID=networkid SKULOC=skuloc PERIOD=period
          MEAN=mean VARIANCE=variance;
run;
```

The HORIZON=8 option specifies the number of periods for which the inventory parameters need to be calculated. The NODE, ARC, and DEMAND statements specify the mapping from variables in the input data sets to variables that are required by the procedure. For example, LEADTIME=LEADTIME in the NODE statement names the variable leadtime in the nodedata1 data set that contains the lead-time information. Such statements are not required if the input data sets use default variable names. In this example, the variables in all three data sets use the default names. Therefore, the MIRP procedure can also be called by the following statements. Default names are listed in the section "Syntax" on page 65.

```
proc mirp nodedata=nodedata1 arcdata=arcdata1 demanddata=demanddata1
          out=out_example1 horizon=8;
run;
```

Figure 3.1 shows the output data set. The reorder and order-up-to levels are given for each period in the planning horizon. To replenish the inventory, the store needs to calculate its inventory position, which is the sum of on-hand inventory and in-transit inventory (that is, shipments yet to be received). If the inventory position is below the reorder level, the store should place an order so that its inventory position is increased to the order-up-to level.

Figure 3.1 Output Data Set of a Single Location

Obs	Network ID	Sku Loc	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	N1	RW	1	1	26	27	7	SUCCESSFUL
2	N1	RW	1	2	26	27	7	SUCCESSFUL
3	N1	RW	1	3	26	27	7	SUCCESSFUL
4	N1	RW	1	4	26	27	7	SUCCESSFUL
5	N1	RW	1	5	26	27	7	SUCCESSFUL
6	N1	RW	1	6	26	27	7	SUCCESSFUL
7	N1	RW	1	7	26	27	7	SUCCESSFUL
8	N1	RW	1	8	26	27	7	SUCCESSFUL

Two-Echelon Distribution Network

Consider a distribution network that has one warehouse and two retailer locations. A finished product *F* is stored at the warehouse *W* with a lead time of three weeks from an external supplier and a unit holding cost of 16 per week. The product is shipped to two retailer locations, *C* and *D*. The unit holding costs at *C* and *D* are 18 per week. The lead time between *W* and *C* is three weeks and between *W* and *D* is four weeks. The required service levels are 95% for both *C* and *D* and 85% for *W*. In this example, the base period is defined as one week.

You can use PROC MIRP to compute replenishment parameters that minimize the inventory holding cost for the entire network while meeting the service-level constraints at all locations.

To solve the inventory replenishment problem, the following input data sets are created. The arcdata2 data set specifies the predecessor and successor of each arc in the network. In contrast with the preceding example, the forecast in the demanddata2 data set fluctuates over the planning horizon.

```
data nodedata2;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3. description $15.
         lt sl hc;
datalines;
N2 FW warehouse       3 0.85 16
N2 FC retailer 1      3 0.95 18
N2 FD retailer 2      4 0.95 18
;

data arcdata2;
   format networkid $2. head tail $8.;
   input networkid $3. head $9. tail $3. qty;
datalines;
N2 EXTERNAL FW 1
N2 FW       FC 1
N2 FW       FD 1
;

data demanddata2;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N2 FC 1 20 25
N2 FC 2 15 16
N2 FC 3 10 9
N2 FC 4 10 9
N2 FC 5 15 16
N2 FC 6 15 16
N2 FC 7 20 25
N2 FC 8 20 25
N2 FD 1 10 9
N2 FD 2 10 9
N2 FD 3 15 16
```

```
N2 FD 4 15 16
N2 FD 5 20 25
N2 FD 6 20 25
N2 FD 7 15 16
N2 FD 8 15 16
;
```

The following call to PROC MIRP computes the inventory policy parameters at each location for the next eight weeks:

```
proc mirp nodedata=nodedata2 arcdata=arcdata2 demanddata=demanddata2
          out=out_example2 horizon=8;
run;
```

The output data set (Figure 3.2) contains reorder and order-up-to levels for every SKU-location at every period. Because the demand at the retailer locations shifts from week to week, these levels also change.

Figure 3.2 Output Data Set of a Two-Echelon Distribution Network

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	N2	FC	retailer 1	1	1	66	67	12.0000	SUCCESSFUL
2	N2	FC	retailer 1	1	2	60	61	11.0000	SUCCESSFUL
3	N2	FC	retailer 1	1	3	60	61	11.0000	SUCCESSFUL
4	N2	FC	retailer 1	1	4	72	73	13.0000	SUCCESSFUL
5	N2	FC	retailer 1	1	5	83	84	14.0000	SUCCESSFUL
6	N2	FC	retailer 1	1	6	89	90	15.0000	SUCCESSFUL
7	N2	FC	retailer 1	1	7	95	96	16.0000	SUCCESSFUL
8	N2	FC	retailer 1	1	8	95	96	16.0000	SUCCESSFUL
9	N2	FD	retailer 2	1	1	81	82	12.0000	SUCCESSFUL
10	N2	FD	retailer 2	1	2	93	94	14.0000	SUCCESSFUL
11	N2	FD	retailer 2	1	3	99	100	15.0000	SUCCESSFUL
12	N2	FD	retailer 2	1	4	99	100	15.0000	SUCCESSFUL
13	N2	FD	retailer 2	1	5	99	100	15.0000	SUCCESSFUL
14	N2	FD	retailer 2	1	6	93	94	14.0000	SUCCESSFUL
15	N2	FD	retailer 2	1	7	87	88	13.0000	SUCCESSFUL
16	N2	FD	retailer 2	1	8	87	88	13.0000	SUCCESSFUL
17	N2	FW	warehouse	2	1	113	114	8.3663	SUCCESSFUL
18	N2	FW	warehouse	2	2	140	141	8.8516	SUCCESSFUL
19	N2	FW	warehouse	2	3	146	147	8.0317	SUCCESSFUL
20	N2	FW	warehouse	2	4	151	152	8.5158	SUCCESSFUL
21	N2	FW	warehouse	2	5	150	151	10.0700	SUCCESSFUL
22	N2	FW	warehouse	2	6	149	150	10.2421	SUCCESSFUL
23	N2	FW	warehouse	2	7	149	150	10.5292	SUCCESSFUL
24	N2	FW	warehouse	2	8	148	149	10.7002	SUCCESSFUL

Two-Echelon Assembly Network

Consider a manufacturing process in a plant P. Two components, A and B, are assembled into finished products, F. Both components are purchased from external suppliers. The lead times for A and B are one period and two periods, respectively. The unit holding cost is 1 per period for A and 2 for B. It takes three units of A and two units of B to produce one unit of F. The assembly process takes three periods. It costs the plant 13 per period to store one unit of F. The required service level is 95% for F, A, and B. The finished product F faces customer demand with a mean of 16 and a variance of 36 per period for the next eight periods.

The total holding cost of the two components that go into the finished products can be computed as follows:

$$\begin{aligned} &\text{Unit holding cost of } A \times \text{Quantity of } A \text{ into } F \\ &+ \text{Unit holding cost of } B \times \text{Quantity of } B \text{ into } F \\ &= 1 \times 3 + 2 \times 2 = 7 \end{aligned}$$

Because the unit holding cost of F is 13, a significant value must be added during the assembly process. This also implies that it is much more expensive to hold the finished goods than to hold the components.

To solve the inventory replenishment problem for the two-echelon assembly network, the following input data sets are created. The arcdata3 data set specifies the bill of material relationship (that is, the quantity) that is assigned between the components and the finished goods.

```
data nodedata3;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3. description $15.
         lt sl hc;
datalines;
N3 AP component 1     1 0.95  1
N3 BP component 2     2 0.95  2
N3 FP finished goods  3 0.95 13
;

data arcdata3;
   format networkid $2. head tail $8.;
   input networkid $3. head $9. tail $3. quantity;
datalines;
N3 EXTERNAL AP 1
N3 EXTERNAL BP 1
N3 AP       FP 3
N3 BP       FP 2
;
```

```
data demanddata3;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N3 FP 1 16 36
N3 FP 2 16 36
N3 FP 3 16 36
N3 FP 4 16 36
N3 FP 5 16 36
N3 FP 6 16 36
N3 FP 7 16 36
N3 FP 8 16 36
;
```

The following call to PROC MIRP computes the inventory policy parameters for the finished goods and each of the components for the next eight periods. The output data set is shown in Figure 3.3.

```
proc mirp nodedata=nodedata3 arcdata=arcdata3 demanddata=demanddata3
          out=out_example3 horizon=8;
run;
```

Figure 3.3 Output Data Set of a Two-Echelon Assembly Network

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	N3	FP	finished goods	1	1	83	84	20.0000	SUCCESSFUL
2	N3	FP	finished goods	1	2	83	84	20.0000	SUCCESSFUL
3	N3	FP	finished goods	1	3	83	84	20.0000	SUCCESSFUL
4	N3	FP	finished goods	1	4	83	84	20.0000	SUCCESSFUL
5	N3	FP	finished goods	1	5	83	84	20.0000	SUCCESSFUL
6	N3	FP	finished goods	1	6	83	84	20.0000	SUCCESSFUL
7	N3	FP	finished goods	1	7	83	84	20.0000	SUCCESSFUL
8	N3	FP	finished goods	1	8	83	84	20.0000	SUCCESSFUL
9	N3	AP	component 1	2	1	131	132	37.7582	SUCCESSFUL
10	N3	AP	component 1	2	2	137	138	40.5700	SUCCESSFUL
11	N3	AP	component 1	2	3	137	138	39.7859	SUCCESSFUL
12	N3	AP	component 1	2	4	137	138	40.8998	SUCCESSFUL
13	N3	AP	component 1	2	5	137	138	40.7839	SUCCESSFUL
14	N3	AP	component 1	2	6	137	138	41.6389	SUCCESSFUL
15	N3	AP	component 1	2	7	134	135	38.9277	SUCCESSFUL
16	N3	AP	component 1	2	8	134	135	38.1062	SUCCESSFUL
17	N3	BP	component 2	2	1	129	130	34.1092	SUCCESSFUL
18	N3	BP	component 2	2	2	131	132	34.6335	SUCCESSFUL
19	N3	BP	component 2	2	3	131	132	34.2036	SUCCESSFUL
20	N3	BP	component 2	2	4	131	132	34.7761	SUCCESSFUL
21	N3	BP	component 2	2	5	131	132	35.4390	SUCCESSFUL
22	N3	BP	component 2	2	6	131	132	35.4613	SUCCESSFUL
23	N3	BP	component 2	2	7	129	130	33.9277	SUCCESSFUL
24	N3	BP	component 2	2	8	129	130	33.1062	SUCCESSFUL

Multiple Networks within a Single Call

The previous examples can be solved in a single call to PROC MIRP by combining their input data. In the following statements, the input data sets of the distribution and assembly networks are combined:

```
data nodedata4;
   set nodedata2 nodedata3;
run;

data arcdata4;
   set arcdata2 arcdata3;
run;

data demanddata4;
   set demanddata2 demanddata3;
run;

proc mirp nodedata=nodedata4 arcdata=arcdata4
          demanddata=demanddata4 out=out_example4
          horizon=8;
run;
```

Distinct network IDs are assigned to each network. Because PROC MIRP computes inventory control parameters for one network at a time, SKU-locations and arcs in the data sets must be grouped by their corresponding networks. In addition, the sequence of networks in the input data sets must be the same. In this example, networks are arranged in the order of N2 and N3 for all input data sets.

Syntax

The following statements are available in the MIRP procedure.

PROC MIRP < *options* > **;**
  NODE / < *options* > **;**
  ARC / < *options* > **;**
  DEMAND / < *options* > **;**
  INVENTORY / < *options* > **;**

If the input data sets use the default variable names, then only the PROC MIRP statement is required. If the input data sets do not use the default variable names, then you must also specify the NODE, ARC, and DEMAND statements. The following sections provide a functional summary of the statements and options you can use in the MIRP procedure, then describe the PROC MIRP statement, and then describe the other statements in alphabetical order.

Functional Summary

Table 3.1 summarizes the statements and options available for the MIRP procedure, classified by function.

Table 3.1 PROC MIRP Functional Summary

Description	Statement	Option
Data Set Specifications		
Input arc data set	PROC MIRP	ARCDATA=
Input demand data set	PROC MIRP	DEMANDDATA=
Input inventory data set	PROC MIRP	INVENTORYDATA=
Output data set for warning or error messages	PROC MIRP	MESSAGE=
Input node data set	PROC MIRP	NODEDATA=
Output data set for results	PROC MIRP	OUT=
Objective Specifications		
Order generation	PROC MIRP	OBJECTIVE=CREATEORDER
Evaluation of internal service level	PROC MIRP	OBJECTIVE=EVALISL
Optimization of internal service level	PROC MIRP	OBJECTIVE=OPTISL
Policy optimization	PROC MIRP	OBJECTIVE=OPTPOLICY
Key performance indicator (KPI) prediction	PROC MIRP	OBJECTIVE=PREDICTKPI
Policy optimization and order generation	PROC MIRP	OBJECTIVE=POLICY_ORDER
Policy optimization, order generation, and KPI prediction	PROC MIRP	OBJECTIVE=POLICY_ORDER_KPI
Optimization of internal service level and policy optimization	PROC MIRP	OBJECTIVE=OPTIMIZATION
Order generation and KPI prediction	PROC MIRP	OBJECTIVE=ORDER_KPI
Miscellaneous Specifications		
Choice of coefficient of variation definition	PROC MIRP	CV2=
Statistical model for demand forecast	PROC MIRP	DEMANDMODEL=
Length of the forecast interval	PROC MIRP	FORECASTINTERVAL=
Length of the planning horizon	PROC MIRP	HORIZON=
Lookup table to be created	PROC MIRP	LOOKUPTABLE=
Maximum allowed coefficient of variation	PROC MIRP	MAXCV=
Maximum number of messages to be printed	PROC MIRP	MAXMESSAGES=
Minimum allowed coefficient of variation	PROC MIRP	MINCV=
Number of networks to be processed	PROC MIRP	NETWORKCNT=
Policy parameter types	PROC MIRP	POLICYPARM=
Number of replications for simulation	PROC MIRP	REPLICATIONS=
SKU-locations to be single-echelon	PROC MIRP	SINGLEECHELON=
Inventory distribution system	PROC MIRP	SYSTEM=
Node Data Set Specifications		
Batch size	NODE	BATCHSIZE=
Inventory capacity	NODE	CAPACITY=

Table 3.1 *continued*

Description	Statement	Option
Demand interval	NODE	DEMANDINTERVAL=
SKU-location description	NODE	DESCRIPTION=
Fixed ordering cost	NODE	FIXEDCOST=
Unit holding cost	NODE	HOLDINGCOST=
Expected lead time	NODE	LEADTIME=
Maximum lead time	NODE	LEADTIMEMAX=
Minimum lead time	NODE	LEADTIMEMIN=
Minimum presentation level	NODE	MPL=
Network ID	NODE	NETWORKID=
Next replenishment period	NODE	NEXTREPLENISH=
Maximum order size	NODE	ORDERMAX=
Minimum order size	NODE	ORDERMIN=
Periods between replenishments	NODE	PBR=
Replenishment policy type	NODE	POLICYTYPE=
Target service level	NODE	SERVICELEVEL=
Service-level type	NODE	SERVICETYPE=
SKU-location ID	NODE	SKULOC=
Arc Data Set Specifications		
Network ID	ARC	NETWORKID=
Unit pipeline cost	ARC	PIPELINECOST=
Predecessor ID	ARC	PREDECESSOR=
Bill of material quantity	ARC	QUANTITY=
Successor ID	ARC	SUCCESSOR=
Demand Data Set Specifications		
Demand average	DEMAND	MEAN=
Network ID	DEMAND	NETWORKID=
Period	DEMAND	PERIOD=
Period description	DEMAND	PERIODDESC=
SKU-location ID	DEMAND	SKULOC=
Demand variance	DEMAND	VARIANCE=
Inventory Data Set Specifications		
Inventory amount	INVENTORY	AMOUNT=
Network ID	INVENTORY	NETWORKID=
Order flag	INVENTORY	ORDERFLAG=
Order-up-to level	INVENTORY	ORDERUPTOLEVEL=
Period	INVENTORY	PERIOD=
Reorder level	INVENTORY	REORDERLEVEL=
SKU-location ID	INVENTORY	SKULOC=

PROC MIRP Statement

PROC MIRP < *options* > **;**

The PROC MIRP statement invokes the MIRP procedure. You can specify the following *options*:

ARCDATA=*SAS-data-set*

names the SAS data set that contains the network specification. Variables in the data set are defined in the section "ARC Statement" on page 72. This option is required unless you specify SINGLEECHELON=YES. When you specify the ARCDATA= option, you must also specify a variable in the NETWORKID= option in the NODE statement and you must group the data by that variable.

CV2=YES | NO

specifies whether PROC MIRP uses the classic definition or a customized definition of the coefficient of variation (CV). You can specify the following values:

YES uses a classic definition of CV, which is the ratio between the standard deviation (the square root of the variance) of demand and the average of demand.

NO uses the customized definition of CV, which is the ratio between the variance and the average of demand.

This option applies only when DEMANDMODEL=DISCRETE. If DEMANDMODEL=CONTINUOUS, the classic definition of CV is used. By default, CV2=NO.

DEMANDDATA=*SAS-data-set*

names the SAS data set that contains forecasts of customer demand. Variables in the data set are defined in the section "DEMAND Statement" on page 73. This option is required, and the data must be grouped by the variable specified in the NETWORKID= option in the NODE statement.

DEMANDMODEL=DISCRETE | CONTINUOUS

specifies how PROC MIRP is to model the demand forecast. You can specify the following values:

DISCRETE models the demand by a set of discrete statistical distributions. More specifically, depending on the ratio between the mean and variance of the demand, the procedure selects from the Bernoulli, binomial, Poisson, negative binomial, or geometric distribution or from any mix of these distributions. This value is recommended for modeling slow-moving items such as spare parts.

CONTINUOUS models the demand by either a normal distribution or a mixed-normal distribution. This value is a better fit for fast-moving items.

By default, DEMANDMODEL=CONTINUOUS. For more information about the demand distributions, see the section "Modeling Demand" on page 82.

FORECASTINTERVAL=*n*

specifies the number of base periods in each forecast period in the DEMANDDATA= data set. For example, if FORECASTINTERVAL=7, one forecast period is equivalent to seven base periods. By default, FORECASTINTERVAL=1.

This option is useful when the forecast period is at a higher granular level than the base period for the inventory replenishment. Quite often the demand forecast is conducted weekly in order to achieve sufficient accuracy, and the inventory is replenished daily. You need to know what daily demand looks like in order to make daily replenishment decisions. To handle the discrepancy in the time scale, you can break down the weekly forecast into a daily forecast (via equal split or through a daily index). PROC MIRP can also break down the forecast, and it uses an equal split.

HORIZON=*n*
specifies the number of base periods for which policy parameters or key performance indicators (KPIs) or both are computed. By default, HORIZON=12.

INVENTORYDATA=*SAS-data-set*
names the SAS data set that contains inventory or policy parameters or both. Variables in this data set are defined in the section "INVENTORY Statement" on page 74. This option is required when you specify one of the following values in the OBJECTIVE= option: CREATEORDER, PREDICTKPI, ORDER_KPI, or POLICY_ORDER_KPI. If you specify this option, you must group the data by the variable specified in the NETWORKID= option in the NODE statement.

LOOKUPTABLE=YES | NO
specifies whether to create a lookup table for fast calculation. You can specify the following values:

YES creates a lookup table. This option can boost the speed of the MIRP procedure when policy optimization is needed for a large number of SKU-locations.

NO does not create a lookup table.

By default, LOOKUPTABLE=NO.

MAXCV=*number*
specifies the maximum coefficient of variation (CV) that is allowed in the demand forecast. The definition of CV depends on the CV2= option if DEMANDMODEL=DISCRETE. When PROC MIRP detects any demand forecast that has a CV that exceeds *number*, it writes a warning message to the log and decreases the variance of the demand to meet *number*. By default, MAXCV=1.

MAXMESSAGES=*n*
specifies the maximum number of warning and error messages that PROC MIRP writes to the log. If this option is omitted, the procedure displays all messages.

MESSAGE=*SAS-data-set*
names the output data set to contain any warning or error messages or both. If this option is omitted, this data set is not created. The variables in this data set are defined in the section "Variables in the MESSAGE= Data Set" on page 87. The data are grouped by the variable specified in the NETWORKID= option in the NODE statement.

MINCV=*number*
specifies the minimum coefficient of variation (CV) that is allowed in the demand forecast. The definition of CV depends on the CV2= option if DEMANDMODEL=DISCRETE. When PROC MIRP detects any demand forecast that has a CV that falls below *number*, it writes a warning message to the log and increases the variance of the demand to meet *number*. By default, MINCV=0.1.

NETWORKCNT=*n*

specifies the number of networks (NETWORKIDs) to be processed. The first *n* NETWORKIDs are processed. By default, all NETWORKIDs are processed.

NODEDATA=*SAS-data-set*

names the SAS data set that contains information about each SKU-location. These variables are defined in the section "NODE Statement" on page 75. This option is required, and the data must be grouped by the variable specified in the NETWORKID= option in the NODE statement.

OBJECTIVE=*option*

specifies how the procedure is used. You can specify one of the following *options*:

OPTPOLICY	optimizes reorder and order-up-to levels for all locations at each planning period, subject to their service-levels constraints.
OPTISL	optimizes service levels of locations that are not customer-facing, subject to service-level constraints at customer-facing locations.
EVALISL	evaluates costs of a network, subject to service-level constraints at all locations.
CREATEORDER	determines order quantities for all locations, based on inventory control policies and current on-hand and pipeline inventory.
PREDICTKPI	estimates key performance indicators (KPIs) for all locations in a network at each planning period, based on inventory control policies and current on-hand and pipeline inventory.
POLICY_ORDER	is equivalent to the combination of OPTPOLICY and CREATEORDER.
POLICY_ORDER_KPI	is equivalent to the combination of OPTPOLICY, CREATEORDER, and PREDICTKPI.
ORDER_KPI	is equivalent to the combination of CREATEORDER and PREDICTKPI.
OPTIMIZATION	is equivalent to the combination of OPTISL and OPTPOLICY.

By default, OBJECTIVE=OPTPOLICY.

OUT=*SAS-data-set*

names the output data set to contain computation results for all SKU-locations. If this option is omitted, PROC MIRP creates a data set and names it according to the DATA*n* naming convention. The variables in this data set are defined in the section "Variables in the OUT= Data Set" on page 83. The data are grouped by the variable specified in the NETWORKID= option in the NODE statement.

POLICYPARM=INTEGER | DOUBLE

specifies the type of reorder and order-up-to levels. You can specify the following values:

INTEGER	specifies that reorder and order-up-to levels are integers. This value is preferred in discrete manufacturing and distribution applications (such as spare parts management).
DOUBLE	specifies that reorder and order-up-to levels are doubles. This value is often appropriate in chemical processes.

By default, POLICYPARM=INTEGER.

REPLICATIONS=*n*

specifies the number of simulation replications to be used in policy optimization and KPI prediction. By default, REPLICATIONS=200. Because PROC MIRP uses simulation in the optimization and prediction steps, the accuracy increases as the number of replications increases. However, the run time also increases. To determine the number of replications needed, you can use the formula

$$n \geq \left(\frac{z \times C_v \times 2}{\delta} \right)^2$$

where n denotes the number of replications, z is the z-value of the confidence level, C_v is the average coefficient of variation of the demand forecast, and δ is the desired accuracy around the average demand. Consider the following example:

- The average coefficient of variation of the demand forecast is 0.45; that is, $C_v = 0.45$.
- The desired accuracy around the average demand is 2.5% in simulation. This is essentially the ratio between the size of the confidence interval around the average demand from the simulation and the average demand in the DEMANDDATA= data set. In this example, $\delta = 0.025$.
- The confidence level is 95%. This sets $z = 1.96$ based on the standard normal distribution.

The number of replications in this example is

$$n \geq (1.96 \times 0.45 \times 2/0.025)^2 \approx 4979$$

SINGLEECHELON=YES | NO

specifies whether SKU-locations that are defined in the NODEDATA= data set are single-echelon. You can specify the following values:

YES specifies that SKU-locations are single-echelon.

NO specifies that SKU-locations are not single-echelon.

By default, SINGLEECHELON=NO.

SYSTEM=PULL | PUSH

specifies whether to push excess inventory to downstream SKU-locations. During a promotional or seasonal sale, for example, you might want to use the push system to transfer excess inventory from a warehouse to retailers even though the retailers already have enough inventory to satisfy forecast demand. You can specify the following values:

PULL requests that PROC MIRP attempt to best satisfy the demand at each SKU-location and that any excess inventory at an upstream SKU-location not be distributed to the downstream SKU-locations.

PUSH requests that all excess inventory at an upstream SKU-location be distributed among the downstream SKU-locations, subject to any capacity constraints.

By default, SYSTEM=PULL. When SYSTEM=PUSH, the variables that are specified in the ORDERMAX= and ORDERMIN= options in the NODE statement are ignored. Instead, you can use the CAPACITY= and BATCHSIZE= options in the NODE statement to specify variables that place bounds on the amount of inventory that can be pushed to a SKU-location.

ARC Statement

ARC /< *options* > **;**

The ARC statement enables you to name variables in the data set that you specify in the ARCDATA= option in the PROC MIRP statement. If you do not specify one of the following options to name a variable, PROC MIRP searches for that variable by its default names. Table 3.2 summarizes the variables in the ARCDATA= data set.

Table 3.2 Variables in ARCDATA= Data Set

Option That Specifies Variable Name	Required	Variable Type	Default Variable Names
NETWORKID=	Yes	Character	NETWORKID, NETID
PIPELINECOST=	No	Numeric	PIPELINECOST, PSCOST
PREDECESSOR=	Yes	Character	PREDECESSOR, HEAD
QUANTITY=	No	Numeric	QUANTITY, QTY
SUCCESSOR=	Yes	Character	SUCCESSOR, TAIL

NETWORKID=*variable*

NETID=*variable*

identifies a character variable in the ARCDATA= data set that specifies the network ID for each arc.

PIPELINECOST=*variable*

PSCOST=*variable*

identifies a numeric variable in the ARCDATA= data set that specifies the unit cost per period of inventory in transit from the predecessor to the successor. Missing values of this variable are assumed to be equal to 0.

PREDECESSOR=*variable*

HEAD=*variable*

identifies a character variable in the ARCDATA= data set that specifies the SKU-location of the predecessor of each arc.

If an arc links a SKU-location to an external supplier, specify PREDECESSOR=EXTERNAL. Note that EXTERNAL is a reserved word and can be used only in this situation.

QUANTITY=*variable*

QTY=*variable*

identifies a numeric variable in the ARCDATA= data set that specifies the bill of material (BOM) quantity between the predecessor and the successor of each arc. It is the number of units at the predecessor required to produce one unit at the successor. Missing values of this variable are assumed to be equal to 1.

SUCCESSOR=*variable*

TAIL=*variable*

identifies a character variable in the ARCDATA= data set that specifies the SKU-location of the successor of each arc.

DEMAND Statement

DEMAND /< *options*> **;**

The DEMAND statement enables you to name variables in the data set that you specify in the DEMAND-DATA= option in the PROC MIRP statement. If you do not specify one of the following options to name a variable, PROC MIRP searches for that variable by its default names. Table 3.3 summarizes the variables in the DEMANDDATA= data set.

Table 3.3 Variables in DEMANDDATA= Data Set

Option That Specifies Variable Name	Required	Variable Type	Default Variable Names
MEAN=	Yes	Numeric	MEAN, AVERAGE
NETWORKID=	Yes	Character	NETWORKID, NETID
PERIOD=	No	Numeric	PERIOD, TIME
PERIODDESC=	No	Numeric	PERIODDESC
SKULOC=	Yes	Character	SKULOC, NODEID, ITEMID
VARIANCE=	Yes	Numeric	VARIANCE, VAR

MEAN=*variable*

AVERAGE=*variable*

identifies a numeric variable in the DEMANDDATA= data set that specifies the average of demand. When demand intervals are specified in the NODEDATA= data set, the mean and variance of positive demand should be provided in the DEMANDDATA= data set.

NETWORKID=*variable*

NETID=*variable*

identifies a character variable in the DEMANDDATA= data set that specifies the network ID for each SKU-location.

PERIOD=*variable*

TIME=*variable*

identifies a numeric variable in the DEMANDDATA= data set that specifies the time period of demand forecast. Valid values are positive integers (such as $1, 2, 3, \ldots$).

PERIODDESC=*variable*

identifies a numeric variable in the DEMANDDATA= data set that specifies the description of the time period of demand forecast as a SAS date.

SKULOC=*variable*

NODEID=*variable*

ITEMID=*variable*

identifies a character variable in the DEMANDDATA= data set that specifies the ID for each SKU-location.

VARIANCE=*variable*

VAR=*variable*

identifies a numeric variable in the DEMANDDATA= data set that specifies the variance of demand. The mean and variance of demand can both be zero. However, a zero mean and a positive variance are not allowed. When demand intervals are specified in the NODEDATA= data set, the mean and variance of positive demand should be provided in the DEMANDDATA= data set.

INVENTORY Statement

INVENTORY */< options>* **;**

The INVENTORY statement enables you to name variables in the data set that you specify in the INVENTORY= option in the PROC MIRP statement. If you do not specify one of the following options to name a variable, PROC MIRP searches for that variable by its default names. Table 3.4 summarizes the variables in the INVENTORYDATA= data set.

The variables that are specified in the ORDERUPTOLEVEL= and REORDERLEVEL= options are not required when the value of the OBJECTIVE= option in the PROC MIRP statement is OPTPOLICY, POLICY_ORDER, POLICY_ORDER_KPI, or OPTIMIZATION. For all other values of the OBJECTIVE= option, these variables are required.

Table 3.4 Variables in INVENTORYDATA= Data Set

Option That Specifies Variable Name	Required	Variable Type	Default Variable Names
AMOUNT=	No	Numeric	AMOUNT, AMT
NETWORKID=	Yes	Character	NETWORKID, NETID
ORDERFLAG=	No	Numeric	ORDERFLAG
ORDERUPTOLEVEL=	Yes/No	Numeric	ORDERUPTOLEVEL, OUTL
PERIOD=	No	Numeric	PERIOD, TIME
REORDERLEVEL=	Yes/No	Numeric	REORDERLEVEL, ROL
SKULOC=	Yes	Character	SKULOC, NODEID, ITEMID

AMOUNT=*variable*

AMT=*variable*

identifies a numeric variable in the INVENTORYDATA= data set that specifies the amount of inventory to arrive at a location for a period. The value of the variable can be negative at period 1 to represent initial backlog. Missing values of this variable are assumed to be equal to 0.

NETWORKID=*variable*

NETID=*variable*

identifies a character variable in the INVENTORYDATA= data set that specifies the network ID for each SKU-location.

ORDERFLAG=*variable*

identifies a numeric variable in the INVENTORYDATA= data set that specifies the order flag at a SKU-location for a period. If the variable is set to 1, a replenishment order can be placed at the SKU-location at a period. Otherwise, no orders can be placed. Missing values of this variable are not allowed. Not using the variable is equivalent to setting the variable to 1 at all SKU-locations for all periods.

ORDERUPTOLEVEL=*variable*

OUTL=*variable*

identifies a numeric variable in the INVENTORYDATA= data set that specifies the order-up-to level at a SKU-location for a period. This option is required when you specify one of the following values in the OBJECTIVE= option in the PROC MIRP statement: CREATEORDER, PREDICTKPI, or ORDER_KPI. Missing values of this variable are assumed to be equal to 0.

PERIOD=*variable*

TIME=*variable*

identifies a numeric variable in the INVENTORYDATA= data set that specifies the arrival period of inventory at a SKU-location.

REORDERLEVEL=*variable*

ROL=*variable*

identifies a numeric variable in the INVENTORYDATA= data set that specifies the reorder level at a SKU-location for a period. This option is required when you specify one of the following values in the OBJECTIVE= option in the PROC MIRP statement: CREATEORDER, PREDICTKPI, or ORDER_KPI. Missing values of this variable are assumed to be equal to 0.

SKULOC=*variable*

NODEID=*variable*

ITEMID=*variable*

identifies a character variable in the INVENTORYDATA= data set that specifies the ID for each SKU-location.

NODE Statement

NODE /< *options*> **;**

The NODE statement enables you to name variables in the data set that you specify in the NODEDATA= option in the PROC MIRP statement. If you do not specify one of the following options to name a variable, PROC MIRP searches for that variable by its default names. Table 3.5 summarizes the variables in the NODEDATA= data set.

Table 3.5 Variables in NODEDATA= Data Set

Option That Specifies Variable Name	Required	Variable Type	Default Variable Names
BATCHSIZE=	No	Numeric	BATCHSIZE, LOTSIZE
CAPACITY=	No	Numeric	CAPACITY
DEMANDINTERVAL=	No	Numeric	DEMANDINTERVAL, DIT
DESCRIPTION=	No	Character	DESCRIPTION, DESC
FIXEDCOST=	No	Numeric	FIXEDCOST, FC
HOLDINGCOST=	Yes	Numeric	HOLDINGCOST, HC
LEADTIME=	Yes	Numeric	LEADTIME, LT
LEADTIMEMAX=	No	Numeric	LEADTIMEMAX, LTMAX
LEADTIMEMIN=	No	Numeric	LEADTIMEMIN, LTMIN
MPL=	No	Numeric	MPL
NETWORKID=	Yes	Character	NETWORKID, NETID
NEXTREPLENISH=	No	Numeric	NEXTREPLENISH, NRP
ORDERMAX=	No	Numeric	ORDERMAX, MAXSIZE
ORDERMIN=	No	Numeric	ORDERMIN, MINSIZE
PBR=	No	Numeric	PBR
POLICYTYPE=	No	Character	POLICYTYPE, POLICY
SERVICELEVEL=	Yes	Numeric	SERVICELEVEL, SL
SERVICETYPE=	No	Character	SERVICETYPE, SLTYPE
SKULOC=	Yes	Character	SKULOC, NODEID, ITEMID

BATCHSIZE=*variable*

LOTSIZE=*variable*

identifies a numeric variable in the NODEDATA= data set that contains the batch-size constraint on the size of orders that can be placed at each SKU-location. For example, if the batch size is 12, the order quantity must be an integer multiple of 12. The value of this variable must be 0 or positive. A value of 0 means that there is no batch-size constraint. Missing values of this variable are assumed to be equal to 0.

CAPACITY=*variable*

identifies a variable that specifies the inventory capacity at a particular SKU-location. If SYSTEM=PULL in the PROC MIRP statement, the variable that is specified in the CAPACITY= option is ignored; you can use the variables that are specified in the ORDERMAX=, ORDERMIN=, and BATCHSIZE= options to place bounds on the order size.

DEMANDINTERVAL=*variable*

DIT=*variable*

identifies a numeric variable in the NODEDATA= data set that contains the average number of periods between two positive demands. For example, the value of 2 means that the demand comes every two periods on average. This variable is used to model intermittent demand. When this variable is specified, the demand mean and variance in the DEMANDDATA= data set must be the mean and variance of positive demand that can occur in a period. The value of this variable must be 1 or greater. The value of 1 means that the demand mean and variance in the DEMANDDATA= data set are for the demand per period. Missing values of this variable are assumed to be equal to 1.

DESCRIPTION=*variable*

DESC=*variable*

identifies a character variable in the NODEDATA= data set that contains the description for each SKU-location. If this variable is specified in the NODEDATA= data set, the information is stored in the DESCRIPTION variable in the output data set. Otherwise, the output data set does not contain the DESCRIPTION variable.

FIXEDCOST=*variable*

FC=*variable*

identifies a numeric variable in the NODEDATA= data set that contains the fixed ordering cost for each SKU-location. The fixed ordering cost is charged when a replenishment order is placed. It is independent of the order amount. Missing values of this variable are assumed to be equal to 0.

HOLDINGCOST=*variable*

HC=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the unit holding cost per period for each SKU-location. This variable is required.

Sometimes it is difficult to obtain unit holding costs. Unit costs and costs of capital are more readily available. For example, if the unit cost of a product is 10 and the cost of capital (also called the opportunity cost of inventory held) is 12% per year, the unit holding cost per month for this product can be estimated as $10 \times 12\%/12 = 0.1$. Other cost elements, such as warehousing costs, are also part of the unit holding cost.

LEADTIME=*variable*

LT=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the expected lead time for each SKU-location. This variable is required, and all values of this variable must be nonnegative integers.

A lead time has two major components:

- Fulfillment-related lead time is the time from the current time until a future period when orders that could not be filled because the on-hand inventory at a location is insufficient to satisfy orders from downstream will be filled. This time is likely to be random.
- Transit-related lead time is the time required to physically deliver an order (full or partial) from one location to another. This time might be random but is less likely to have variation than fulfillment-related lead time.

When you calculate lead time, you should consider only transit-related lead time as the input into the MIRP procedure. Fulfillment-related lead time is taken into account implicitly by the procedure (more specifically, by incorporating backlogs into the optimization).

You can also specify a maximum lead time and a minimum lead time for each SKU-location by using the LEADTIMEMAX= and LEADTIMEMIN= options, respectively. The distribution of the lead time for each SKU-location is determined as follows:

- When the minimum lead time < the expected lead time < the maximum lead time, a triangular distribution is used to model the lead-time uncertainty.

- When the minimum lead time = the expected lead time < the maximum lead time, or the minimum lead time < the expected lead time = the maximum lead time, a uniform distribution between the minimum lead time and the maximum lead time is used to model the lead-time uncertainty.
- When the minimum lead time = the expected lead time = the maximum lead time, the lead time is considered constant.
- When the minimum lead time and the maximum lead time are not specified, the lead time is considered constant.

LEADTIMEMAX=*variable*

LTMAX=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the maximum lead time for each SKU-location. All values of this variable must be nonnegative integers or missing values. Missing values are assumed to be equal to the expected lead time.

If a maximum lead time variable is specified in the NODEDATA= data set, then a minimum lead time variable must also be specified by using the LEADTIMEMIN= option. For information about how PROC MIRP uses the minimum lead time and maximum lead time to determine the distribution of lead time, see the LEADTIME= option.

LEADTIMEMIN=*variable*

LTMIN=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the minimum lead time for each SKU-location. All values of this variable must be nonnegative integers or missing values. Missing values are assumed to be equal to the expected lead time.

If a minimum lead time variable is specified in the NODEDATA= data set, then a maximum lead time variable must also be specified by using the LEADTIMEMAX= option. For information about how PROC MIRP uses the minimum lead time and maximum lead time to determine the distribution of lead time, see the LEADTIME= option.

MPL=*variable*

identifies a variable that specifies a lower bound on the order-up-to level determined during policy optimization for each SKU-location. This option is used only when the value of the OBJECTIVE= option in the PROC MIRP statement is OPTPOLICY, POLICY_ORDER, POLICY_ORDER_KPI, or OPTIMIZATION. Missing values of this variable are not allowed.

NETWORKID=*variable*

NETID=*variable*

identifies a character variable that specifies the network ID for each SKU-location in the NODEDATA= data set. This variable enables you to compute policy parameters for multiple networks by issuing a single call to PROC MIRP, which is illustrated in the section "Multiple Networks within a Single Call" on page 65. This variable is required.

NEXTREPLENISH=*variable*

NRP=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies when the first replenishment order can be made for each SKU-location. For example, if the value of the variable is equal to 1, the first replenishment order can be placed in period 1. If the value of the variable is 4, no replenishment order can be placed until period 4. Missing values of this variable are assumed to be equal to 1.

ORDERMAX=*variable*

MAXSIZE=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the maximum amount of an order that can be placed at each SKU-location. Missing values of this variable are assumed to be the largest double that your operating system supports. If SYSTEM=PUSH in the PROC MIRP statement, the variable that is specified in the ORDERMAX= option is ignored; you can use the variable that is specified in the CAPACITY= option to place bounds on the order size.

ORDERMIN=*variable*

MINSIZE=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the minimum amount of an order that can be placed at each SKU-location. Missing values of this variable are assumed to be equal to 0. If SYSTEM=PUSH in the PROC MIRP statement, the variable that is specified in the ORDERMIN= option is ignored; you can use the variable that is specified in the BATCHSIZE= option to place bounds on the order size.

PBR=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the number of periods between two replenishment orders for each SKU-location. Missing values of this variable are assumed to be equal to 1.

When you specify this option and the variable specified by the POLICYTYPE= variable indicates the base-stock policy, replenishment orders can be placed only once every number of periods specified by the PBR= variable. For example, if the value of the PBR= variable is 4 and the value of the POLICYTYPE= variable is BS to indicate the base-stock policy, then replenishment orders can be placed only once every four periods.

When you specify this option and the variable specified by the POLICYTYPE= variable indicates the min-max policy, replenishment orders can be placed in any period. However, the policy parameters are computed such that on average the number of periods between two replenishment orders is not less than the value of the PBR= variable.

POLICYTYPE=*variable*

POLICY=*variable*

identifies a character variable in the NODEDATA= data set that specifies the policy type to be used at each SKU-location. The procedure supports two policy types: BS (base-stock policy) and SS (min-max policy). Missing values of this variable are assumed to be BS.

The base-stock policy is also called one-to-one replenishment policy, because the order-up-to level is equal to the reorder level plus 1. This policy is recommended when the fixed ordering cost is insignificant compared to the inventory holding cost.

The min-max policy is recommended when the fixed ordering cost is significantly higher than the inventory holding cost. In this case, a large amount of inventory should be ordered so that replenishment is less frequent. The order-up-to level in this policy is greater than the reorder level by at least 1. When the difference is exactly 1, this policy is reduced to the base-stock policy.

SERVICELEVEL=*variable*

SL=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the service-level requirement for each SKU-location. This variable is required.

SERVICETYPE=*variable*

SLTYPE=*variable*

identifies a numeric variable in the NODEDATA= data set that specifies the type of service level used for each SKU-location. The procedure supports three service types: RR (ready rate), FR (fill rate), and BR (back-order ratio). Missing values of this variable are assumed to be RR.

The ready rate is also called the non-stockout probability. It is the probability of not running out of stock (that is, having positive on-hand inventory) at the end of a period. A historical ready rate can be measured as the ratio between the number of periods that have positive on-hand inventory and the total number of periods under consideration. The fill rate is the percentage of demand that is satisfied immediately by on-hand inventory. The back-order ratio is the ratio between the average backlog at the end of a period and the average demand during the period.

SKULOC=*variable*

NODEID=*variable*

ITEMID=*variable*

identifies a character variable in the NODEDATA= data set that specifies the ID for each SKU-location. This variable is required.

Details

Service-Level Analysis

The service-level analysis sets service-level targets at all locations in a supply chain network. From the analysis perspective, there are two types of locations: locations that face customer demand directly (called *customer-facing locations*) and locations that face replenishment orders from other locations within the network (called *internal locations*).

Service-level targets at customer-facing locations are set based on many factors, such as customers' expectations of service as perceived by companies that provide the service, competitors' service-level targets, available budgets for inventory, and so on. Companies usually know what service levels to set at customer-facing locations, and they do not change the targets very often (maybe once a year). Example 3.13 shows how you can use the MIRP procedure to understand the trade-off between service levels and costs.

Most companies set their service-level targets at internal locations to be the same as or similar to those at customer-facing locations. The likely reason for this is that they manage locations independently from one another (*decentralized management*) without considering the interdependency between locations. Management teams are graded based on the inventory performance at locations under their management. Under decentralized control, total network costs are not minimized.

You can specify the OBJECTIVE=EVALISL option in the PROC MIRP statement to evaluate the costs of any service-level settings (Example 3.20). When you specify OBJECTIVE=OPTISL, PROC MIRP can also determine the optimal service-level settings for all internal locations in a network (Example 3.21). Service-level targets at internal locations are seldom reset unless significant changes have occurred. For example, the targets might be reset once every quarter because of the demand seasonality. Therefore, when you specify OBJECTIVE=OPTISL or OBJECTIVE=EVALISL, PROC MIRP makes the following assumptions:

- Demand forecast is stationary.
- Base-stock policy is used.
- Batch-size constraints, minimum-order constraints, and maximum-order constraints are short-term constraints and are therefore ignored.
- Lead time is constant. Thus, minimum lead time and maximum lead time are ignored.

Policy Optimization

As discussed in the preceding section, service levels at customer-facing locations are usually set for a year, and service levels at internal locations are usually set for a quarter. After these service levels are decided, you can calculate optimal reorder and order-up-to levels for all products at all locations in your supply chain network. This process is called policy optimization, which is supported by the OBJECTIVE=OPTPOLICY option in PROC MIRP.

Policy optimization is done for a short-term planning horizon, anywhere from a few days to a few weeks. Because the demand forecast changes from one day to the next, or from one week to the next, you need to determine the optimal reorder and order-up-to levels for each period in your planning horizon.

Because policy optimization is very close to the execution of the inventory replenishment, PROC MIRP takes account of nonstationary demand, ordering constraints, and lead time variations. It also supports a min-max policy. These assumptions are quite different from the assumptions that are made in the service-level analysis.

Most of the time, only the demand forecast changes from one period to the next. Therefore, policy optimization is aligned with the demand forecast. You do not need to reoptimize policy parameters if there is no change in the demand forecast.

Order Generation

After reorder and order-up-to levels are determined, you can use the following steps to calculate how much to order for each product at each location:

1. Calculate total inventory, which is the sum of current on-hand inventory and the pipeline inventory (also called *in-transit inventory*).
2. If the total inventory is less than or equal to the reorder level at the current period, you need to place a replenishment order.
3. The amount of the order is equal to the difference between the order-up-to level and the total inventory.
4. If there are any order constraints, revise the order amount accordingly.

Order generation is usually conducted daily, but you do not have to place replenishment orders every day.

The OBJECTIVE=CREATEORDER option in the PROC MIRP statement is designed to support order generation. When multiple locations are ordering from one upstream location and the total order amount exceeds the available inventory at the upstream location, you have to decide how to allocate the available inventory among all orders. PROC MIRP uses marginal analysis to decide the best allocation based on service-level targets, demand volume, and current inventory status at the downstream locations. (See Example 3.23.)

KPI Prediction

Given the current on-hand and in-transit inventory, you want to know some key performance indicators in the future. Such KPIs include service levels, backlogs, on-hand inventory, in-transit inventory, and so on. Knowing KPI projections helps you uncover potential problems in the future and take proper actions to resolve them ahead of time.

The OBJECTIVE=PREDICTKPI option in the PROC MIRP statement projects a list of KPIs for a specified set of policy parameters and current inventory status. The procedure uses simulation for the projection. The accuracy of the projection increases with a larger number of simulation replications, but at the expense of a longer run time.

Modeling Demand

In the DEMANDDATA= data set, each demand forecast is defined by two variables: mean and variance. However, most forecasting systems provide a predicted value and a standard error instead. The mean is equivalent to the predicted value, and the variance is equal to the standard error squared.

To optimize inventory policy parameters, the MIRP procedure fits the demand forecast to a statistical distribution that has exactly the same mean and variance, as follows:

- When you specify DEMANDMODEL=CONTINUOUS in the PROC MIRP statement, PROC MIRP uses a normal distribution or a mixed normal distribution.
- When you specify DEMANDMODEL=DISCRETE in the PROC MIRP statement, PROC MIRP chooses a Bernoulli distribution, a binomial distribution, a Poisson distribution, a negative binomial distribution, a geometric distribution, or a mix of these distributions. The choice is made such that the mean and variance of the chosen distribution are as close as possible to the demand mean and demand variance, respectively, that are specified in the DEMANDDATA= data set.

Variables in the OUT= Data Set

The OUT= data set contains the following variables.

STATUS
contains the calculation status. This variable has the following possible values:

- SUCCESSFUL
- INVD_VALUE, which means that data errors are detected in a network
- OUT_OF_MEM, which means that the procedure cannot obtain sufficient memory

AllocatedQuantity
contains the allocated quantity from supplying locations. When a SKU-location replenishes inventory from an external supplier, the allocated quantity is equal to the suggested order quantity. If it replenishes from another location within the same network, the allocated quantity might be less than the suggested quantity. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: CREATEORDER or POLICY_ORDER.

BacklogMean
contains the average backlog at the end of a period. The backlog is the amount of demand that is not satisfied by the on-hand inventory at a SKU-location. The backlog is carried over to future periods until it is satisfied. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

BacklogVar
contains the variance of backlog at the end of a period. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

BackorderRatio
contains the back-order ratio at a SKU-location. The back-order ratio is the ratio between the average backlog in a period and the average demand in the period. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTIMIZATION, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

DelayMean
contains the average delivery delay from supplying locations. If a SKU-location replenishes from another SKU-location within the same network, the replenishment order might not be completely satisfied because of insufficient inventory at the supplying location. The remaining portion of the order is to be delivered in future, causing a delay in the delivery. The variable measures the average of such delays. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL or OPTISL.

DelayVar
contains the variance of delivery delay from supplying locations. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL or OPTISL.

Description
contains description information about a SKU-location. This is the same information as in the variable in the NODEDATA= data set that is named by the DESCRIPTION= option.

Echelon
contains the echelon level of a SKU-location. If a SKU-location does not have a successor, its echelon level is 1. If a SKU-location has one or more successors, its echelon level is the maximum of the echelon levels of all its successors plus 1.

ExternalDemandMean
contains the average of customer demand. This is the same as the demand mean in the DEMANDDATA= data set. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

ExternalDemandVar
contains the variance of customer demand. This is the same as the demand variance in the DEMANDDATA= data set. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

FillRate
contains the fill rate. The fill rate is the percentage of demand that is immediately satisfied by on-hand inventory. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTIMIZATION, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

InternalDemandMean
contains the average of internal transfer orders at a SKU-location. The internal transfer orders are the replenishment orders from the immediate successors of the SKU-location. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

InternalDemandVar
contains the variance of internal transfer orders. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

NetworkID
contains the network ID. The value is the same as the value of the variable in the NODEDATA= data set that is named by the NETWORKID= option.

OhHoldingCost
contains the holding cost of on-hand inventory at the end of a period. This is equal to the product of the unit holding-cost and the on-hand inventory at the end of the period (that is, OnHandMean). This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL or OPTISL.

OnHandMean

contains the average on-hand inventory at the end of a period. This is used to compute the holding cost of on-hand inventory at the end of the period (that is, OhHoldingCost). This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

OnHandVar

contains the variance of on-hand inventory at the end of a period. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

OrderMean

contains the average order quantity at a SKU-location for each period in the planning horizon. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

OrderQuantity

contains the order quantity at period 1. The value of this variable is calculated based on the current inventory (on-hand and in-transit inventory), the reorder and order-up-to levels, and the order constraints. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: CREATEORDER or POLICY_ORDER.

OrderUpToLevel

contains the order-up-to level. This is the target inventory level used to generate orders. Orders are generated so that the total inventory is equal to or higher than order-up-to levels. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTIMIZATION, OPTISL, OPTPOLICY, POLICY_ORDER, or POLICY_ORDER_KPI.

OrderVar

contains the variance of the order quantity at a SKU-location for each period in the planning horizon. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

Period

contains the time period ID. The variable contains sequential numbers from 1 to the last period of the planning horizon.

PeriodDesc

contains the date description of a period. The value is the same as the value of the variable in the DEMANDDATA= data set that is named by the PERIODDESC= option.

PipelineCost

contains the total cost of pipeline inventory. The pipeline cost is equal to the average pipeline inventory (that is, PipelineMean) multiplied by the unit pipeline cost in the ARCDATA= data set. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL or OPTISL. If unit pipeline cost does not exist in the ARCDATA= data set, the MIRP procedure does not produce PipelineCost in the OUT= data set.

PipelineMean
contains the average pipeline inventory. The average pipeline inventory is the sum of average demand over the expected lead time. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

PipelineVar
contains the variance of pipeline inventory. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

PlannedReceiptMean
contains the average projected receipts at a SKU-location for each period in the planning horizon. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

PlannedReceiptVar
contains the variance of the projected receipts at a SKU-location for each period in the planning horizon. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

ReadyRate
contains the ready rate. The ready rate is the probability of having positive on-hand inventory at any particular period. It is also called non-stockout probability. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTIMIZATION, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

ReceiptQuantity
contains the receipt quantity in a period. A receipt quantity in a period is the sum of scheduled receipts (which are specified in the INVENTORY= data set) and orders that are newly generated and are to be delivered in that period. This variable is included when the OBJECTIVE= POLICY_ORDER is specified in the PROC MIRP statement.

ReorderLevel
contains the reorder level. When the total inventory (that is, the sum of on-hand and in-transit inventory) drops to or below the reorder level, a replenishment order should be generated. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: OPTIMIZATION, OPTPOLICY, POLICY_ORDER, or POLICY_ORDER_KPI.

SafetyStock
contains the amount of safety stock. Safety stock is used to cover demand uncertainty from downstream locations and lead-time uncertainty from upstream locations. If there is no uncertainty, there is no need to keep safety stock. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTIMIZATION, OPTISL, OPTPOLICY, POLICY_ORDER, or POLICY_ORDER_KPI.

ShortfallMean
contains the average shortfall at a SKU-location for each period in the planning horizon. Shortfall is the backlog from upstream locations. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

ShortfallVar

contains the variance of the shortfall at a SKU-location for each period in the planning horizon. This variable is included when one of the following values is specified in the OBJECTIVE= option in the PROC MIRP statement: EVALISL, OPTISL, ORDER_KPI, POLICY_ORDER_KPI, or PREDICTKPI.

SkuLoc

contains the name of a SKU-location at which a data issue is detected.

Variables in the MESSAGE= Data Set

The MESSAGE= data set contains the following variables.

Dataset

contains the name of the data set in which a data issue is detected.

Message

contains the message text of a data issue.

MessageNo

contains the internal message number of a data issue.

Msg_SK

contains the message key of a data issue. The message key is a sequential number starting from 1.

NetworkID

contains the ID of a network in which a data issue is detected. The value is the same as the value of the variable in the NODEDATA= data set that is named by the NETWORKID= option.

Period

contains the time period in which a data issue is detected.

Predecessor

contains the SKU-location ID of a predecessor of an arc in the ARCDATA= data set where a data issue is detected.

SkuLoc

contains the name of a SKU-location at which a data issue is detected.

Successor

contains the SKU-location ID of a successor of an arc in the ARCDATA= data set where a data issue is detected.

Memory Requirement

To estimate memory requirement (in megabytes) for a network, you can use the formula

$$(2868 + 484 \times P + 4 \times R + 16 \times L + 28 \times P \times R) \times L \div (1024)^2$$

where P is the number of periods in the planning horizon, L is the number of SKU-locations in a network, and R is the number of replications. This estimate is an upper bound on the memory requirement.

For example, for a network that contains 150 SKU-locations, a planning horizon of 52 periods, and 5,000 replications, you need memory in the amount of

$$(2868 + 484 \times 52 + 4 \times 5000 + 16 \times 150 + 28 \times 52 \times 5000) \times 150 \div (1024)^2 \approx 1048.7$$

This is a little more than 1GB of memory.

Examples

Example 3.1: Policy Optimization in a Single-Location Network

This example demonstrates policy optimization with the MIRP procedure in a single-location network. The structure of three input data sets is discussed, and the safety stock calculation is explained.

The following DATA step reads the key information about a product at a warehouse into the nodedata_single data set. This is the simplest setting for a multi-echelon inventory optimization: the network contains one node. The replenishment lead time from an external supplier to the location is one period. The holding cost is 5 per unit, per period. The target service level is 90%, which is measured based on the ready rate (RR). The base-stock policy, which is denoted as BS, is used.

```
/* Node data set for single-location network */
data nodedata_single;
     infile datalines dlm=',';
     input   networkid:$2.
            skuloc: $3.
            description: $16.
            leadTime
            holdingCost
            serviceLevel
            serviceType: $2.
            policyType: $2.;
datalines;
S1, W,  WAREHOUSE, 1,  5,  0.9, RR, BS
;
```

The following arcdata_single data set describes the network structure that supports the product. Because it is a single-location problem, there is only one arc: from EXTERNAL to W.

```
/* Arc data set for single-location network */
data arcdata_single;
    infile datalines dlm=',';
    input   networkid: $2.
            predecessor: $8.
            successor: $3.;
datalines;
S1, EXTERNAL, W
;
```

The demand forecast is stored in the following demanddata_single data set. For the next five periods, the demand has an average of 90 and a variance of 900. In this example, the demand is stationary (that is, the mean and variance are the same for all periods). In practice, nonstationary demand is more common.

```
/* Demand data set for single-location network */
data demanddata_single;
    infile datalines dlm=',';
    input   networkid: $2.
            skuloc: $3.
            period
            mean
            variance;
datalines;
S1, W, 1, 90, 900
S1, W, 2, 90, 900
S1, W, 3, 90, 900
S1, W, 4, 90, 900
S1, W, 5, 90, 900
;
```

The following call to the MIRP procedure calculates optimal reorder and order-up-to levels. In inventory theory, reorder and order-up-to levels are also called parameters for the inventory control policy (or policy parameters for short). The OBJECTIVE=OPTPOLICY option requests that the procedure calculate optimal policy parameters for the location. The HORIZON=5 option requests that parameters be generated for five periods.

```
title 'Single-Location Network';
title2 'Policy Optimization';
proc mirp nodedata=nodedata_single arcdata=arcdata_single
          demanddata=demanddata_single out=optpolicy_single
          HORIZON=5 OBJECTIVE=OPTPOLICY;
run;
```

Output 3.1.1 contains reorder and order-up-to levels for each period in the five-period planning horizon. Because the demand is stationary, the levels are the same throughout the horizon.

Safety stock is needed to cover demand variation. In this example, you need to carry inventory to cover demand for two periods, because the lead time is one and the base-stock policy is used. So on average, the total demand to cover is 180 units. The inventory that is needed in addition to these 180 units is safety stock (that is, $235 - 180 = 55$).

Output 3.1.1 Policy Optimization: Single-Location Network

Single-Location Network
Policy Optimization

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	234	235	55	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	234	235	55	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	234	235	55	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	234	235	55	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	234	235	55	SUCCESSFUL

Example 3.2: Order Generation in a Single-Location Network

This example explains how a replenishment order is triggered and how the amount of the order is calculated in a single-location network.

After you calculate policy parameters in Example 3.1, you want to decide how much to order based on the current inventory at the location. In the following DATA step, you create an inventory data set that contains the reorder and order-up-to levels from the previous call to PROC MIRP. You also specify that there are 96 units of the inventory at the beginning of the first period and there is no planning receipt in the future periods.

```
/* Inventory data set for single-location network */
data inventorydata_single;
   set optpolicy_single;
   if period=1 then amount=96;
   else amount=.;
run;
```

The following call to the MIRP procedure uses the nodedata_single, arcdata_single, and demanddata_single data sets from Example 3.1 and the inventorydata_single data set. The OBJECTIVE=CREATEORDER option requests that the procedure suggest an order quantity.

```
title2 'Order Generation';
proc mirp nodedata=nodedata_single arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single
          out=createorder_single horizon=5 OBJECTIVE=CREATEORDER;
run;
```

In Output 3.2.1, the order quantity is 139 units. This quantity is derived as follows:

1. The total inventory, including the beginning on-hand inventory at the first period and the planned receipts in the future, is 96 units.

2. The total inventory is less than the reorder level of 234 units; therefore a replenishment order is placed.

3. The order amount is the difference between the order-up-to level and the current total inventory (that is, $235 - 96 = 139$).

The allocated quantity in the output data set is the amount that the location received from its supplier, which happens to be an external supplier in this example. By assumption, all external suppliers are always able to deliver, and therefore the allocated quantity is equal to the order quantity. In a multi-echelon network, where a location might order from another location in the network, it is quite likely that the allocated quantity is less than the order quantity when the supplying location does not have sufficient inventory. For more information, see Example 3.23.

Output 3.2.1 Order Generation: Single-Location Network

Single-Location Network
Order Generation

Obs	Network ID	Sku Loc	Description	Echelon	Period	Order Quantity	Allocated Quantity	_STATUS_
1	S1	W	WAREHOUSE	1	1	139	139	SUCCESSFUL

Example 3.3: KPI Prediction in a Single-Location Network

This example demonstrates how you can use the MIRP procedure to predict key performance indicators (KPI) for given on-hand and in-transit inventory at a location.

In Example 3.2, a replenishment order in the amount of 139 units is placed. Because the lead time from the external supplier is one period, a delivery of 139 units is received at period 2. The following call to the SQL procedure adds the delivery to the inventorydata_single data set in period 2:

```
proc sql noprint;
   select AllocatedQuantity into: AllocAmt from createorder_single;
   update inventorydata_single set amount=&AllocAmt where period=2;
quit;
```

Now you have 96 units of on-hand inventory at period 1, and you receive 139 units at period 2. The following call to the MIRP procedure estimates KPIs. Note that a new option is added: REPLICATIONS=5000, which requests that the procedure run 5,000 simulation replications to estimate the KPIs. The accuracy of the estimates increases with the number of replications, but at the expense of longer run time.

```
title2 'KPI Prediction';
proc mirp nodedata=nodedata_single arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single
          OUT=predictkpi_single horizon=5 OBJECTIVE=PREDICTKPI
          REPLICATIONS=5000;
run;
```

Output 3.3.1 shows that the planned receipts are 96 and 139 for the first two periods, which are equal to the value of the AMOUNT variable in the inventorydata_single data set. Their variances are zero, because these amounts are fixed. Also, the order mean and variance are zero at period 1 because the total inventory at period 1 ($96 + 139 = 235$) is equal to the order-up-to level at period 1. So there is no need to place another replenishment order.

The ready rate at period 1 is quite low, because there are only 96 units of inventory for the period and the demand has a mean of 90 units and a variance of 900 units. Significant backlog is projected for period 1. The ready rates from period 2 onward are very close to the target service level of 90%. This is expected, because the optimal reorder and order-up-to levels are used and any backlogs that occurred at period 1 are satisfied in later periods.

The average order quantity from periods 2 through 5 is $(90.3378 + 89.9639 + 89.3397 + 90.1216)/4 = 89.94075$, which is very close to the average demand of 90 during those periods. Theoretically, they should be exactly equal in the example. They are different because simulation is used in the estimation.

The average variance of order quantities from periods 2 through 5 is $(917.5825 + 925.177 + 891.304 + 917.221)/4 = 912.821125$, which is also very close to the average variance of demand during those periods. They should also be equal in theory. The variance of order quantity can be significantly higher than the variance in demand when you use order constraints or the min-max policy. This is called the bullwhip effect. Example 3.19 demonstrates this effect with order constraints.

Output 3.3.1 KPI Prediction: Single-Location Network

Single-Location Network
KPI Prediction

Obs	Network ID	Sku Loc	Description	Echelon	Period	External Demand Mean	External Demand Var	Internal Demand Mean	Internal Demand Var
1	S1	W	WAREHOUSE	1	1	90	900	0	0
2	S1	W	WAREHOUSE	1	2	90	900	0	0
3	S1	W	WAREHOUSE	1	3	90	900	0	0
4	S1	W	WAREHOUSE	1	4	90	900	0	0
5	S1	W	WAREHOUSE	1	5	90	900	0	0

Obs	Order Mean	Order Var	Planned Receipt Mean	Planned Receipt Var	Pipeline Mean	Pipeline Var	OnHand Mean	OnHand Var	Backlog Mean
1	0.0000	0.000	96.000	0.000	139.000	0.000	15.0955	395.38	9.43364
2	90.3378	917.582	139.000	0.000	90.338	917.582	56.6423	1544.42	1.94400
3	89.9639	925.177	90.338	917.582	89.964	925.177	57.4567	1591.12	1.76038
4	89.3397	891.304	89.964	925.177	89.340	891.304	57.3119	1547.10	1.77337
5	90.1216	917.221	89.340	891.304	90.122	917.221	57.1314	1548.43	2.00373

Obs	Backlog Var	Shortfall Mean	Shortfall Var	Ready Rate	Fill Rate	Backorder Ratio	_STATUS_
1	237.270	0	0	0.56231	0.89557	0.10443	SUCCESSFUL
2	61.301	0	0	0.89954	0.97839	0.02161	SUCCESSFUL
3	53.870	0	0	0.90489	0.98030	0.01970	SUCCESSFUL
4	57.150	0	0	0.90547	0.98032	0.01968	SUCCESSFUL
5	64.418	0	0	0.90002	0.97767	0.02233	SUCCESSFUL

Example 3.4: Combined Objective Options

The previous three examples call the MIRP procedure to perform one specific task. You can also use combined objective options to request that the procedure do more than one task.

The following call requests that PROC MIRP determine the replenishment quantity and then project KPIs based on the replenishment decision:

```
title2 'order_kpi';
proc mirp nodedata=nodedata_single arcdata=arcdata_single
          demanddata=demanddata_single
          inventorydata=inventorydata_single
          OUT=orderkpi_single horizon=5 objective=order_kpi
          REPLICATIONS=5000;
run;
```

The output of this PROC MIRP call is exactly the same as Output 3.3.1. The only difference is that this example combines the order quantity generation and KPI prediction in a single call to PROC MIRP.

If you want to calculate an optimal policy and then determine the order quantity based on current inventory status, you can use the OBJECTIVE=POLICY_ORDER option, as follows:

```
data inventorydata_single;
    infile datalines dlm=',';
    input  networkid: $2.
           skuloc: $3.
           period
           amount;
datalines;
S1, W, 1, 120
;

title2 'policy_order';
proc mirp nodedata=nodedata_single arcdata=arcdata_single
          demanddata=demanddata_single
          inventorydata=inventorydata_single
          OUT=policyorder_single horizon=5 objective=policy_order;
run;
```

If you also want to see the projected KPIs based on the optimal policy and the replenishment in period 1, you can request that PROC MIRP do all three tasks by using the OBJECTIVE=POLICY_ORDER_KPI option, as follows:

```
title2 'policy_order_kpi';
proc mirp nodedata=nodedata_single arcdata=arcdata_single
          demanddata=demanddata_single
          inventorydata=inventorydata_single
          OUT=policyorderkpi_single horizon=5 objective=policy_order_kpi;
run;
```

Example 3.5: Batch-Size Constraint in a Single-Location Network

This example demonstrates how a batch-size constraint is modeled using the MIRP procedure and how the constraint might affect policy parameters, suggested order quantities, and KPIs.

The batch-size constraint requests that the size of all replenishment orders be integer multiples of the batch size. For example, for a batch size of 20, the order amount can be only 0, 20, 40, and so on. The constraint is added in the nodedata_single_batchsize data set. First, you calculate optimal reorder and order-up-to levels based on this constraint, as follows:

```
/* Add batch-size constraint */
data nodedata_single_batchsize;
   set nodedata_single;
   batchsize=20;
run;

title2 'Policy Optimization';
title3 'Batch-Size Constraint=20';
proc mirp nodedata=nodedata_single_batchsize arcdata=arcdata_single
         demanddata=demanddata_single OUT=optpolicy_single_batchsize
         horizon=5 objective=optpolicy;
run;
```

As shown in Output 3.5.1, the reorder and order-up-to levels are 224 and 225, respectively. When there is no batch-size constraint, the levels are 234 and 235, as in Output 3.1.1. To understand why the constraint decreases the optimal inventory levels, you need to know all the possible inventory positions that can be reached with and without the constraint. When the constraint does not exist, you can replenish as little as one unit, so you can always hit the target inventory level of 235 units. If you use 234 as the reorder level and use the batch-size constraint, then you overshoot the target level of 235 most of the time. For example, if the total inventory is 234 units, you need to replenish. You need only one unit, but you have to order 20 units because of the constraint. So the total inventory after the replenishment is 254 units. Similarly, if the total inventory is 233 units, you need only two units, but you have to order 20 units, bringing the total inventory to 253 units. The total inventory after the replenishment can be anywhere from 235 to 254. From Example 3.1, you know that you can meet the customer demand with a 90% ready rate at an inventory level of 235 units. Clearly, when the inventory level is 236 units or higher, the service level is higher than 90%. Therefore, the average service level across all possible inventory positions is higher than 90%. In order to achieve the target service level, you have to decrease the reorder and order-up-to levels.

Output 3.5.1 Policy Optimization: Single-Location Network with Batch-Size Constraint

Single-Location Network
Policy Optimization
Batch-Size Constraint=20

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	224	225	45	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	224	225	45	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	224	225	45	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	224	225	45	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	224	225	45	SUCCESSFUL

The following call to the MIRP procedure determines how much to order, given that you have 96 units on hand with no delivery beyond period 1:

```
/* Inventory data set for single-location network */
data inventorydata_single;
   set optpolicy_single_batchsize;
   if period=1 then amount=96;
   else amount=.;
run;

title2 'Order Generation';
title3 'Batch-Size Constraint=20';
proc mirp nodedata=nodedata_single_batchsize arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single
          out=createorder_single_batchsize horizon=5 objective=createorder;
run;
```

Because the total inventory at period 1 is 96 units (which is lower than the reorder level of 224), you have to replenish. The order-up-to level is 225, so you need 129 units. However, because of the batch-size constraint, you end up ordering 140. This is shown in Output 3.5.2.

Output 3.5.2 Order Generation: Single-Location Network with Batch-Size Constraint

```
                              Single-Location Network
                                  Order Generation
                              Batch-Size Constraint=20

      Network  Sku                                        Order    Allocated
Obs     ID     Loc  Description  Echelon  Period  Quantity   Quantity    _STATUS_

 1      S1      W    WAREHOUSE      1        1       140        140      SUCCESSFUL
```

With 96 units on hand and 140 units to be received in period 2, you can estimate KPIs as follows:

```
proc sql noprint;
   select AllocatedQuantity into: AllocAmt from createorder_single_batchsize;
   update inventorydata_single set amount=&AllocAmt where period=2;
quit;

title2 'KPI Prediction';
title3 'Batch-Size Constraint=20';
proc mirp nodedata=nodedata_single_batchsize arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single
          out=predictkpi_single_batchsize horizon=5 objective=predictkpi
          replications=5000;
run;
```

In Output 3.5.3, the projected ready rate for the next five periods and the projected backlog show patterns very similar to those in Output 3.3.1: low ready rate and significant backlog in period 1, close to target service level and much less backlog from period 2 onward. The average order quantity from periods 2 through 5 is also close to 90 units. The difference is the average variance of order quantity for the same periods. It is now $(953.431 + 990.883 + 948.071 + 991.395)/4 = 970.945$. This is about 8% higher than the average variance of demand. In other words, the external supplier is experiencing a higher variance of demand (orders) from the warehouse than the variance of customer demand at the warehouse. The increase in variance is caused by the batch-size constraint. This is commonly called the bullwhip effect.

Output 3.5.3 KPI Prediction with Batch-Size Constraint: Single-Location Network

Single-Location Network
KPI Prediction
Batch-Size Constraint=20

Obs	Network ID	Sku Loc	Description	Echelon	Period	External Demand Mean	External Demand Var	Internal Demand Mean	Internal Demand Var
1	S1	W	WAREHOUSE	1	1	90	900	0	0
2	S1	W	WAREHOUSE	1	2	90	900	0	0
3	S1	W	WAREHOUSE	1	3	90	900	0	0
4	S1	W	WAREHOUSE	1	4	90	900	0	0
5	S1	W	WAREHOUSE	1	5	90	900	0	0

Obs	Order Mean	Order Var	Planned Receipt Mean	Planned Receipt Var	Pipeline Mean	Pipeline Var	OnHand Mean	OnHand Var	Backlog Mean
1	0.000	0.000	96.000	0.000	140.000	0.000	15.0955	395.38	9.43364
2	89.400	953.431	140.000	0.000	89.400	953.431	57.5438	1555.67	1.84550
3	89.812	990.883	89.400	953.431	89.812	990.883	57.5942	1605.54	1.83564
4	89.472	948.071	89.812	990.883	89.472	948.071	57.2957	1565.19	1.84689
5	90.060	991.395	89.472	948.071	90.060	991.395	57.2472	1570.52	2.07681

Obs	Backlog Var	Shortfall Mean	Shortfall Var	Ready Rate	Fill Rate	Backorder Ratio	_STATUS_
1	237.270	0	0	0.56231	0.89557	0.10443	SUCCESSFUL
2	57.884	0	0	0.90354	0.97949	0.02051	SUCCESSFUL
3	56.916	0	0	0.90408	0.97945	0.02055	SUCCESSFUL
4	59.679	0	0	0.90046	0.97951	0.02049	SUCCESSFUL
5	67.666	0	0	0.89541	0.97686	0.02314	SUCCESSFUL

Example 3.6: Minimum-Order Constraint

This example imposes a constraint on the lower bound of the order amount; that is, the order amount must be at least 90 units. This is called a minimum-order constraint. No batch-size constraint is used in this example.

```
/* Add minimum-order constraint */
data nodedata_single_ordermin;
   set nodedata_single;
   ordermin=90;
run;

title2 'Policy Optimization';
title3 'Minimum-Order Constraint = 90';
proc mirp nodedata=nodedata_single_ordermin arcdata=arcdata_single
          demanddata=demanddata_single out=optpolicy_single_ordermin
          horizon=5 objective=optpolicy;
run;
```

As shown in Output 3.6.1, the optimal reorder and order-up-to levels with the minimum-order constraint are lower than those without the constraint shown in Output 3.1.1. Like the batch-size constraint that is discussed in Example 3.5, the minimum-order constraint causes the total inventory to range between the order-up-to level and the order-up-to level plus the minimum order minus one. If you set the order-up-to level without considering this constraint, you can end up with a service level much higher than your target.

Output 3.6.1 Policy Optimization: Single-Location Network with Minimum-Order Constraint

Single-Location Network
Policy Optimization
Minimum-Order Constraint = 90

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	218	219	39	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	218	219	39	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	218	219	39	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	218	219	39	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	218	219	39	SUCCESSFUL

Example 3.7: Maximum-Order Constraint

This example demonstrates how to model the maximum-order constraint in PROC MIRP and demonstrates the impact of this constraint on policy optimization and order generation.

If there is an upper bound on how much supplying nodes can deliver at each time, you can model it by using the maximum-order constraint. In this example, the upper bound is 120 units. No minimum-order or batch-size constraint is used.

```
/* Add maximum-order constraint */
data nodedata_single_ordermax;
   set nodedata_single;
   ordermax=120;
run;

title2 'Policy Optimization';
title3 'Maximum-Order Constraint = 120';
proc mirp nodedata=nodedata_single_ordermax arcdata=arcdata_single
          demanddata=demanddata_single out=optpolicy_single_ordermax
          horizon=5 objective=optpolicy;
run;
```

Output 3.7.1 shows that the reorder and order-up-to levels are higher than those in Example 3.1, where such a constraint does not exist. The constraint introduces a so-called shortfall into the replenishment process. You might need 130 units to reach the order-up-to level, but you have to settle for 120 units because of the constraint. The 10 units are the shortfall. Additional safety stock is needed to cover the potential shortfall at each replenishment; this leads to higher reorder and order-up-to levels.

Output 3.7.1 Policy Optimization: Single-Location Network with Maximum-Order Constraint

Single-Location Network
Policy Optimization
Maximum-Order Constraint = 120

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	241	242	62	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	241	242	62	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	241	242	62	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	241	242	62	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	241	242	62	SUCCESSFUL

In the following DATA step, you set on-hand inventory to be 96 units at period 1 and no delivery at period 2 and beyond. Clearly, you need to replenish at the amount of $242 - 96 = 146$ units. Because you can order at most 120 units, the order quantity is 120 units, as shown in Output 3.7.2.

```
/* Inventory data set for single-location network */
data inventorydata_single;
   set optpolicy_single_ordermax;
   if period=1 then amount=96;
   else amount=.;
run;

title2 'Order Generation';
title3 'Maximum-Order Constraint = 120';
proc mirp nodedata=nodedata_single_ordermax arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single
          out=createorder_single_ordermax horizon=5 objective=createorder;
run;
```

Output 3.7.2 Order Generation: Single-Location Network with Maximum-Order Constraint

Single-Location Network
Order Generation
Maximum-Order Constraint = 120

Obs	Network ID	Sku Loc	Description	Echelon	Period	Order Quantity	Allocated Quantity	_STATUS_
1	S1	W	WAREHOUSE	1	1	120	120	SUCCESSFUL

Example 3.8: Fill Rate

This example demonstrates how to model fill rate as the service-level measure in the MIRP procedure and demonstrates the impact of fill rate on the optimal policy parameters.

Service levels can be measured in several ways. PROC MIRP supports three measures: ready rate, fill rate, and back-order ratio. Example 3.1 uses the ready rate measure to calculate optimal reorder and order-up-to levels. The following data set changes the measure to the fill rate. Optimal levels are recalculated.

```
/* Fill rate measure */
data nodedata_single_fillrate;
   set nodedata_single;
   servicetype="FR";
run;

title2 'Policy Optimization';
title3 'Fill Rate';
proc mirp nodedata=nodedata_single_fillrate arcdata=arcdata_single
          demanddata=demanddata_single out=optpolicy_single_fillrate
          horizon=5 objective=optpolicy;
run;
```

Output 3.8.1 shows that the reorder and order-up-to levels are lower than those for the ready rate measure. This is true in general (but not always), because the fill rate measure is less stringent than the ready rate.

Output 3.8.1 Policy Optimization: Single-Location Network with Fill Rate

Single-Location Network
Policy Optimization
Fill Rate

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	200	201	21	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	200	201	21	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	200	201	21	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	200	201	21	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	200	201	21	SUCCESSFUL

Example 3.9: Random Lead Time

This example demonstrates how PROC MIRP models random lead times and demonstrates the impact of such randomness on optimal policy parameters.

The MIRP procedure assumes that any external suppliers are always able to deliver the exact amount that is ordered. However, the delivery lead time can vary. This can also be true for internal locations in a network. To model the randomness in lead time, the procedure uses three parameters: minimum lead time, expected lead time, and maximum lead time. Recall that in Example 3.1 the expected lead time is 1. The following DATA step sets the minimum lead time to 0 and the maximum lead time to 2. A triangular distribution is used to model the random lead time, where the expected lead time is taken to be the mode of the distribution.

```
/* Random lead time */
data nodedata_single_leadtime;
   set nodedata_single;
   leadtimemin=0;
   leadtimemax=2;
run;

title2 'Policy Optimization';
title3 'Random Lead Time';
proc mirp nodedata=nodedata_single_leadtime arcdata=arcdata_single
         demanddata=demanddata_single out=optpolicy_single_leadtime
         horizon=5 objective=optpolicy;
run;
```

Obviously, the randomness in lead time adds more uncertainty to the amount of inventory available to meet customer demand. To cover the additional uncertainty, more safety stock is required. This is very clear in Output 3.9.1.

Output 3.9.1 Policy Optimization: Single-Location Network with Random Lead Time

Single-Location Network
Policy Optimization
Random Lead Time

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	294	295	115	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	294	295	115	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	294	295	115	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	294	295	115	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	294	295	115	SUCCESSFUL

Example 3.10: Min-Max Policy

This example demonstrates the impact of the min-max policy on policy optimization, order generation, and KPI prediction.

The previous examples use the base-stock policy, which allows a replenishment amount as small as one unit. Such a replenishment strategy is not cost-effective when, regardless of the order amount, a significant cost (or time) exists at the time of replenishment. Under such circumstances, it might be optimal to order more each time but order less frequently. To achieve that, you can use the batch-size constraint or the minimum-order constraint. You can also use another replenishment strategy, the min-max policy.

The following DATA step changes the policy type to SS (which stands for the min-max policy). You can also set the periods between replenishment (PBR) to 2, which means you want to replenish once every two periods on average. With this setting, you can still have consecutive orders if you need to. But in the long run, the average number of periods between replenishments is 2. If you do not set the PBR, its default value is 1, which means you want to order every period. That is equivalent to the base-stock policy.

```
/* Min-max policy */
data nodedata_single_minmax;
   set nodedata_single;
   policytype='SS';
   pbr=2;
run;

title2 'Policy Optimization';
title3 'Min-Max Policy';
proc mirp nodedata=nodedata_single_minmax arcdata=arcdata_single
          demanddata=demanddata_single out=optpolicy_single_minmax
          horizon=5 objective=optpolicy;
run;
```

Output 3.10.1 shows that the difference between the optimal reorder level and the order-up-to level is much greater than 1, which is the characterization of the min-max policy. With this policy, the minimum order quantity is $311 - 181 = 130$ units, which leads to one replenishment in two periods on average.

Output 3.10.1 Policy Optimization: Single-Location Network with Min-Max Policy

Single-Location Network
Policy Optimization
Min-Max Policy

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	181	311	131	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	181	311	131	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	181	311	131	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	181	311	131	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	181	311	131	SUCCESSFUL

The following DATA step specifies the on-hand inventory to be 200 units at period 1 and no delivery in the future periods. So the total inventory at period 1 is 200 units, which is higher than the reorder level of 181. No replenishment is needed, as shown in Output 3.10.2.

```
/* Inventory data set for single-location network */
data inventorydata_single;
   set optpolicy_single_minmax;
   if period=1 then amount=200;
   else amount=.;
run;

title2 'Order Generation';
title3 'Min-Max Policy';
proc mirp nodedata=nodedata_single_minmax arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single
          out=createorder_single_minmax horizon=5 objective=createorder;
run;
```

Output 3.10.2 Order Generation: Single-Location Network with Min-Max Policy

Single-Location Network
Order Generation
Min-Max Policy

Obs	Network ID	Sku Loc	Description	Echelon	Period	Order Quantity	Allocated Quantity	_STATUS_
1	S1	W	WAREHOUSE	1	1	0	0	SUCCESSFUL

The min-max policy enables you to balance fixed ordering costs against inventory holding costs. However, there is a serious side effect: many key performance indicators might vary greatly from one period to another. This is quite clear in Output 3.10.3 after the MIRP procedure is called to project KPIs. Because you set PBR to 2, large orders occur in periods 2 and 4, separated by much smaller orders. Because the lead time is 1, you receive huge deliveries in periods 3 and 5, resulting in an inventory pileup in these two periods. The service levels also fluctuate greatly over the five periods. It is also important to point out that the variance of order quantity is much higher than the variance of customer demand. This is the bullwhip effect caused by the min-max policy. The following call to PROC MIRP projects KPIs for the min-max policy:

```
title2 'KPI Prediction';
title3 'Min-Max Policy';
proc mirp nodedata=nodedata_single_minmax arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single
          out=predictkpi_single_minmax horizon=5 objective=predictkpi
          replications=5000;
run;
```

Output 3.10.3 KPI Prediction: Single-Location Network with Min-Max Policy

Single-Location Network
KPI Prediction
Min-Max Policy

Obs	Network ID	Sku Loc	Description	Echelon	Period	External Demand Mean	External Demand Var	Internal Demand Mean	Internal Demand Var
1	S1	W	WAREHOUSE	1	1	90	900	0	0
2	S1	W	WAREHOUSE	1	2	90	900	0	0
3	S1	W	WAREHOUSE	1	3	90	900	0	0
4	S1	W	WAREHOUSE	1	4	90	900	0	0
5	S1	W	WAREHOUSE	1	5	90	900	0	0

Obs	Order Mean	Order Var	Planned Receipt Mean	Planned Receipt Var	Pipeline Mean	Pipeline Var	OnHand Mean	OnHand Var	Backlog Mean
1	0.000	0.00	200.000	0.00	0.000	0.00	109.662	917.52	0.00000
2	199.653	1389.91	0.000	0.00	199.653	1389.91	28.689	988.15	8.99080
3	16.213	2275.70	199.653	1389.91	16.213	2275.70	130.182	1990.74	0.17109
4	142.697	6750.95	16.213	2275.70	142.697	6750.95	60.220	2498.93	4.11759
5	51.281	7048.95	142.697	6750.95	51.281	7048.95	110.582	3231.99	1.53271

Obs	Backlog Var	Shortfall Mean	Shortfall Var	Ready Rate	Fill Rate	Backorder Ratio	_STATUS_
1	0.000	0	0	1.00000	1.00000	0.000000	SUCCESSFUL
2	321.865	0	0	0.67841	0.90006	0.099938	SUCCESSFUL
3	9.092	0	0	0.99359	0.99809	0.001915	SUCCESSFUL
4	157.495	0	0	0.83517	0.95432	0.045689	SUCCESSFUL
5	69.404	0	0	0.94610	0.98292	0.017078	SUCCESSFUL

Example 3.11: PBR and Next-Replenishment Constraints

This example demonstrates how to use the periods between replenishments (PBR) and next-replenishment constraints to control the replenishment frequency under the base-stock policy.

As shown in Example 3.10, the min-max policy can control how frequently you replenish. You can also control the frequency when using the base-stock policy. The following DATA step uses the base-stock policy with two constraints, PBR=2 and NEXTREPLENISH=2, which mean that you can replenish once every two periods and the earliest time you can do that is period 2. In other words, you can order at period 2, and then period 4, and so on, but you cannot order at periods 1, 3, 5, and so on. With the min-max policy, you can place replenishment orders in consecutive periods if you need to, as long as the average number of periods between replenishment is equal to the PBR. With the base-stock policy and PBR constraints, you can place replenishment orders only at certain periods.

```
/* PBR and next-replenishment constraints */
data nodedata_single_pbr;
   set nodedata_single;
   pbr=2;
   nextreplenish=2;
run;

title2 'Policy Optimization';
title3 'PBR and Next-Replenishment Constraints';
proc mirp nodedata=nodedata_single_pbr arcdata=arcdata_single
         demanddata=demanddata_single out=optpolicy_single_pbr
         horizon=5 objective=optpolicy;
run;
```

Output 3.11.1 shows the reorder and order-up-to levels that are calculated for periods 2 and 4. For the periods in which you cannot replenish, these levels are missing. Because you cannot order in every period, these levels are much higher than those in Example 3.1.

Output 3.11.1 Policy Optimization: Single-Location Network with PBR and Next-Replenishment Constraints

Single-Location Network
Policy Optimization
PBR and Next-Replenishment Constraints

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	.	.	.	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	372	373	103	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	.	.	.	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	372	373	103	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	.	.	.	SUCCESSFUL

Example 3.12: Order-Flag Constraint

This example demonstrates how to use the order-flag constraint to control replenishment in specific periods.

Under the base-stock policy, you can control the replenishment frequency by using PBR constraints. However, you can use PBR constraints to model the frequency only if it has a fixed pattern, such as every two periods. If there is no such pattern, you can use the order-flag constraint. The following DATA step introduces the ORDERFLAG variable to the inventorydata_single_orderflag data set. When the order flag is 1, you can replenish; when it is 0, you cannot replenish. Because you can assign order flags by periods, you can model any arbitrary replenishment pattern.

```
   /* Add order-flag constraint */
   data inventorydata_single_orderflag;
       infile datalines dlm=',';
       input   networkid: $2.
               skuloc: $3.
               period
            orderflag;
   datalines;
   S1, W, 1, 1
   S1, W, 2, 0
   S1, W, 3, 0
   S1, W, 4, 1
   S1, W, 5, 0
   ;

title2 'Policy Optimization';
title3 'Order-Flag Constraint';
proc mirp nodedata=nodedata_single arcdata=arcdata_single
          demanddata=demanddata_single inventorydata=inventorydata_single_orderflag
          out=optpolicy_single_orderflag horizon=5 objective=optpolicy;
run;
```

Output 3.12.1 shows the reorder and order-up-to levels only for the periods in which replenishments are allowed.

Output 3.12.1 Policy Optimization: Single-Location Network with Order-Flag Constraint

Single-Location Network
Policy Optimization
Order-Flag Constraint

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	S1	W	WAREHOUSE	1	1	396	397	37	SUCCESSFUL
2	S1	W	WAREHOUSE	1	2	.	.	.	SUCCESSFUL
3	S1	W	WAREHOUSE	1	3	.	.	.	SUCCESSFUL
4	S1	W	WAREHOUSE	1	4	328	329	59	SUCCESSFUL
5	S1	W	WAREHOUSE	1	5	.	.	.	SUCCESSFUL

Example 3.13: Analysis of Customer Service Level

This example demonstrates how to find an optimal service level at a customer-facing location by using the OBJECTIVE=EVALISL option.

In all the preceding examples, the service level is set at 90%. You might want to know whether this is the right service level. If you lower the service level, you lose more sales revenue because of more backorders, but you carry less inventory. If you raise the service level, you can satisfy more demand, but you have to carry more inventory. To find the optimal service level, you need to balance the inventory holding cost with the back-order costs. The OBJECTIVE=EVALISL option in the MIRP procedure can help you find the perfect balance.

Because setting the service level is a long-term decision, you need to make the decision based on a long-term demand forecast. The following demanddata2_single data set contains such a forecast:

```
data demanddata2_single;
    infile datalines dlm=',';
    input   networkid: $2.
            skuloc: $3.
            mean
            variance;
datalines;
S1, W, 90, 900
;
```

The following macro calls the MIRP procedure repeatedly, with ready rates ranging from 10% to 99%. Costs that are associated with each ready rate are collected in the ReadyRate_Impact data set. The inventory holding cost is calculated as the product of the average on-hand inventory (onhandmean) and the unit holding cost of $5. The back-order cost is computed as the product of the average backlog (backlogmean) and the unit back-order cost of $15. The back-order costs might be the discount that you have to give to customers because of the inventory shortage, or they could be the lost sales revenue. The total cost is the sum of the inventory holding and back-order costs.

```
%macro ReadyRate_Cost_Tradeoff();
options nonotes;
   %let rr=0.1;
   %do %while(%sysevalf(&rr<0.9901));
      proc sql;
         update nodedata_single
            set servicelevel=&rr;
      quit;

      proc mirp nodedata=nodedata_single arcdata=arcdata_single
                demanddata=demanddata2_single OUT=evalisl_single
                objective=EVALISL policyparm=double;
      run;
```

```
      proc sql;
         create table tmp_rr as
            select readyrate,
                        onhandmean*5 as holdingcost label='Holding Cost',
                        backlogmean*15 as backordercost label='Back-Order Cost',
                        (onhandmean*5+backlogmean*15) as totalcost label='Total Cost'
                from evalisl_single;
      quit;

      %if %sysevalf(&rr<0.101) %then %do;
         data ReadyRate_Impact;
            set tmp_rr;
         run;
      %end;
      %else %do;
         proc append base=ReadyRate_Impact data=tmp_rr;run;
      %end;

      %let rr=%sysevalf(&rr+0.01);
   %end;
options notes;
%mend;
%ReadyRate_Cost_Tradeoff;
```

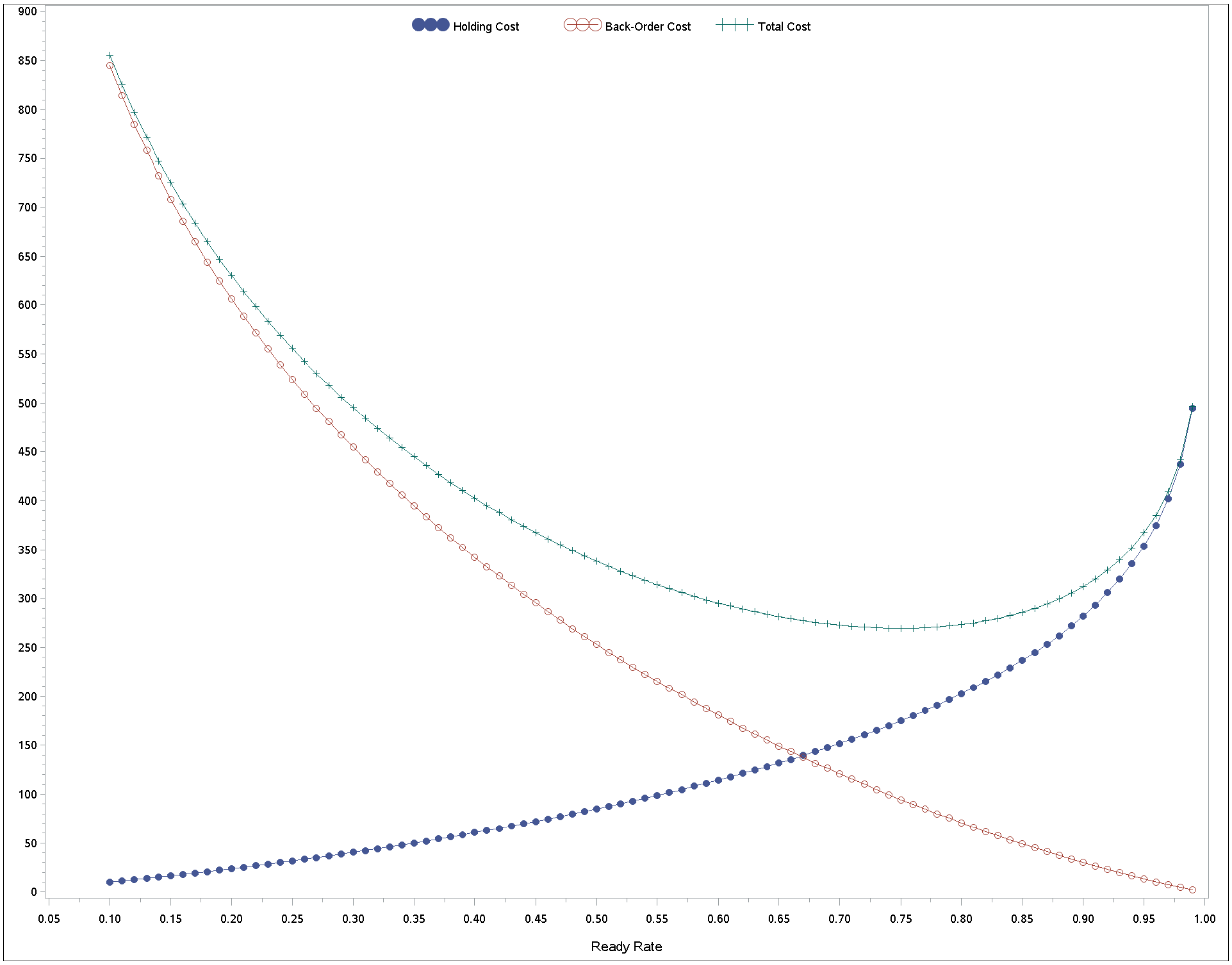
The following GPLOT procedure plots all three costs against the ready rate:

```
goptions reset=all border;
symbol1 interpol=join value=dot color=_style_;
symbol2 interpol=join value=circle color=_style_;
symbol3 interpol=join value=plus color=_style_;
axis1 order=(0.05 to 1 by 0.05)
      label=('Ready Rate');
axis2 order=(0 to 900 by 50)
      label=none;
legend1 label=none
        shape=symbol(4,2)
        position=(top center inside)
        mode=share;
proc gplot data=ReadyRate_impact;
    plot holdingcost*readyrate
        backordercost*readyrate
        totalcost*readyrate
        /overlay legend=legend1
         haxis=axis1 vaxis=axis2;
   run;
quit;
```

Output 3.13.1 shows the following:

- As the ready rate increases, the back-order cost drops.
- As the ready rate increases, the inventory holding cost rises.
- The optimal service level is 75%, which yields the minimum total cost.

Output 3.13.1 Service-Level Impact: Single-Location Network



Example 3.14: Policy Optimization in a Serial Network

This example demonstrates the policy optimization by using PROC MIRP for a serial network.

The following arcdata_serial data set describes the structure of a serial network. It has two locations: location *W*, which replenishes from an external supplier, and location *R*, which orders from *W*.

```
/* Arc data set for two-echelon serial network */
data arcdata_serial;
    infile datalines dlm=',';
    input   networkid: $6.
            predecessor: $8.
            successor: $2.;
datalines;
SERIAL, EXTERNAL, W
SERIAL, W, R
;
```

The following nodedata_serial data set contains key information about the two locations. The target ready rates are 95% for both locations. The lead time from the external supplier to *W* is two periods, and the lead time between *W* and *R* is one period. The unit holding cost at *W* is slightly higher than the unit holding cost at *R*. It is generally true that unit holding costs increase (or do not decrease) as inventory moves from vendor-facing locations to customer-facing locations. In this example, *W* is a vendor-facing location and *R* is a customer-facing location.

```
/* Node data set for two-echelon serial network */
data nodedata_serial;
    infile datalines dlm=',';
    input   networkid:$6.
            skuloc: $2.
            description: $16.
            leadTime
            holdingCost
            serviceLevel
            serviceType: $2.;
datalines;
SERIAL, W, WAREHOUSE, 2, 6.8, 0.95, RR
SERIAL, R, RETAILER 1,1, 7.0, 0.95, RR
;
```

The following demanddata_serial data set contains the forecast of customer demand at location *R*. The forecast varies slightly over the next five periods. This is called nonstationary demand, which is common in practice.

```
/* Demand data set for two-echelon serial network */
data demanddata_serial;
    infile datalines dlm=',';
    input   networkid: $6.
            skuloc: $2.
            period
            mean
            variance;
datalines;
SERIAL, R, 1, 9,  9
SERIAL, R, 2, 11, 11
SERIAL, R, 3, 10, 10
SERIAL, R, 4, 11, 11
SERIAL, R, 5, 9,  9
;
```

The following call to the MIRP procedure calculates the optimal reorder and order-up-to levels for these two locations for the next five periods:

```
title 'Two-Echelon Serial Network';
title2 'Policy Optimization';
proc mirp nodedata=nodedata_serial arcdata=arcdata_serial
          demanddata=demanddata_serial out=optpolicy_serial
          objective=optpolicy  replications=5000 horizon=5;
run;
```

Output 3.14.1 shows the optimal policy parameters for both locations. Because the demand forecast is nonstationary, the parameters vary from one period to the next. The parameters at *W* are more than 10 units higher than those at *R*. This is largely because of the longer lead time at *W*.

Output 3.14.1 Policy Optimization: Two-Echelon Serial Network

Two-Echelon Serial Network
Policy Optimization

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	SERIAL	R	RETAILER 1	1	1	27	28	8.0000	SUCCESSFUL
2	SERIAL	R	RETAILER 1	1	2	28	29	8.0000	SUCCESSFUL
3	SERIAL	R	RETAILER 1	1	3	28	29	8.0000	SUCCESSFUL
4	SERIAL	R	RETAILER 1	1	4	27	28	8.0000	SUCCESSFUL
5	SERIAL	R	RETAILER 1	1	5	25	26	8.0000	SUCCESSFUL
6	SERIAL	W	WAREHOUSE	2	1	41	42	9.9992	SUCCESSFUL
7	SERIAL	W	WAREHOUSE	2	2	40	41	11.0417	SUCCESSFUL
8	SERIAL	W	WAREHOUSE	2	3	39	40	11.0610	SUCCESSFUL
9	SERIAL	W	WAREHOUSE	2	4	37	38	11.0845	SUCCESSFUL
10	SERIAL	W	WAREHOUSE	2	5	37	38	10.9564	SUCCESSFUL

Example 3.15: Order Generation in a Serial Network

As discussed in Example 3.2, external suppliers are assumed to be completely reliable: they can deliver whatever is ordered. You cannot make such an assumption when a location replenishes from another location within a multi-echelon network. The inventory availability at supplying locations must be considered. This example demonstrates the order generation in a serial network.

In the following inventorydata_serial data set, on-hand and future deliveries are specified. At location *R*, 10 units of inventory are on hand in period 1, but no future delivery is planned. At location *W*, on-hand inventory in period 1 is 16 units, and there is one scheduled delivery of 14 units in period 2. The data set also contains the optimal policy parameters from Example 3.14.

```
/* Inventory data set for two-echelon serial network */
data inventorydata_serial;
   set optpolicy_serial;
   if skuloc='R' then do;
         if period=1 then amount=10;
         else amount=.;
      end;
   else do;
            if period=1 then amount=16;
            else if period=2 then amount=14;
            else amount=.;
         end;
run;
```

Next, PROC MIRP is called to determine how much to order at each location. Here you specify that 5,000 simulation replications be used in order to obtain accurate long-run approximations of inventory availability; this was not necessary in the preceding single-echelon examples because those models assumed that sufficient supply was always available to satisfy demand.

```
title2 'Order Generation';
proc mirp nodedata=nodedata_serial arcdata=arcdata_serial
          demanddata=demanddata_serial inventorydata=inventorydata_serial
          out=createorder_serial objective=createorder
           replications=5000 horizon=5;
run;
```

Output 3.15.1 shows that the suggested order quantity is 18 units at *R* and the allocated quantity is only 16 units. These two numbers are derived as follows:

1. The total inventory, which is the sum of on-hand inventory and future deliveries, is 10 units for location *R* in period 1.
2. A replenishment order is triggered because the total inventory is less than the reorder level at location *R*.
3. The amount of the replenishment order at *R* is $28 - 10 = 18$.
4. Because *W* has 16 units of on-hand inventory in period 1, *R* receives only 16 units, which is 2 units fewer than what is needed.

The total inventory is $16 + 14 = 30$ at *W*, so *W* needs 12 units. Because *W* orders from an external supplier, it receives 12 units.

Output 3.15.1 Order Generation: Two-Echelon Serial Network

Two-Echelon Serial Network
Order Generation

Obs	Network ID	Sku Loc	Description	Echelon	Period	Order Quantity	Allocated Quantity	_STATUS_
1	SERIAL	R	RETAILER 1	1	1	18	16	SUCCESSFUL
2	SERIAL	W	WAREHOUSE	2	1	12	12	SUCCESSFUL

Example 3.16: KPI Prediction in a Serial Network

This example demonstrates how to use PROC MIRP to project KPIs in a serial network. It also shows how the starting inventory at an upstream location affects the service-level performance at a downstream location.

In the following call to the SQL procedure, the allocated quantity from the createorder_serial data set is added to the inventorydata_serial data set for locations *R* and *W*. Because the lead time between *R* and *W* is one period, the allocated quantity of *R* is to be received in period 2. The allocated quantity of *W* is to be delivered in period 3 because the lead time at *W* is two periods. Because the replenishment decisions have been made for period 1, there is no need to make the decision again in the KPI prediction. The order-flag constraints are added to ensure this.

```
proc sql noprint;
   select AllocatedQuantity into :AllocAmt1 - :AllocAmt2
        from createorder_serial;
   alter table inventorydata_serial
      add OrderFlag num;
   update inventorydata_serial
        set amount=&AllocAmt1 where skuloc='R' and period=2;
   update inventorydata_serial
        set amount=&AllocAmt2 where skuloc='W' and period=3;
   update inventorydata_serial
        set amount=0 where skuloc='W' and period=1;
   update inventorydata_serial
        set OrderFlag= case when period=1 then 0 else 1 end;
quit;

title2 'KPI Prediction';
proc mirp nodedata=nodedata_serial arcdata=arcdata_serial
          demanddata=demanddata_serial inventorydata=inventorydata_serial
          out=predictkpi_serial objective=predictkpi
           replications=5000 horizon=5;
run;
```

Output 3.16.1 contains projected KPIs for both locations. Because not enough inventory is on hand in period 1, the ready rate at *R* is much lower than the target level of 95%. The ready rate of *R* in period 2 is also lower because *R* receives only 16 units from *W*, 2 units fewer than what it needs. Clearly, the inventory availability at *W* has a major impact on *R*. The ready rate at *R* picks up from period 3 onward as the inventory at *W* moves to ideal levels under the control of the optimal policy.

At *W*, the projected service levels are missing for period 1, because *R* does not order in that period as a result of the order-flag constraint. The ready rate is quite low for periods 2 and 3 because of the low starting inventory. But it gets closer to the targets in period 4 and beyond. The internal demand at *W* is essentially the order quantity from *R*. That is why the mean and variance of the internal demand at *W* are equal to the mean and variance, respectively, of the order quantity at *R*.

Output 3.16.1 KPI Prediction: Two-Echelon Serial Network

Two-Echelon Serial Network
KPI Prediction

Obs	Network ID	Sku Loc	Description	Echelon	Period	External Demand Mean	External Demand Var	Internal Demand Mean	Internal Demand Var
1	SERIAL	R	RETAILER 1	1	1	9	9	0.0000	0.0000
2	SERIAL	R	RETAILER 1	1	2	11	11	0.0000	0.0000
3	SERIAL	R	RETAILER 1	1	3	10	10	0.0000	0.0000
4	SERIAL	R	RETAILER 1	1	4	11	11	0.0000	0.0000
5	SERIAL	R	RETAILER 1	1	5	9	9	0.0000	0.0000
6	SERIAL	W	WAREHOUSE	2	1	0	0	0.0000	0.0000
7	SERIAL	W	WAREHOUSE	2	2	0	0	12.0338	9.1752
8	SERIAL	W	WAREHOUSE	2	3	0	0	10.9953	11.3207
9	SERIAL	W	WAREHOUSE	2	4	0	0	8.9291	9.9268
10	SERIAL	W	WAREHOUSE	2	5	0	0	9.0145	11.2048

Obs	Order Mean	Order Var	Planned Receipt Mean	Planned Receipt Var	Pipeline Mean	Pipeline Var	OnHand Mean	OnHand Var	Backlog Mean
1	0.0000	0.0000	10.0000	0.0000	16.0000	0.0000	1.7452	4.4875	0.77899
2	12.0338	9.1752	16.0000	0.0000	11.5750	5.8023	6.1563	17.4136	0.18564
3	10.9953	11.3207	11.5750	5.8023	10.7637	7.5880	7.7173	20.8576	0.10174
4	8.9291	9.9268	10.7637	7.5880	9.5113	10.8700	7.5085	20.9749	0.14186
5	9.0145	11.2048	9.5113	10.8700	9.0773	11.1445	7.9776	19.5466	0.07472
6	0.0000	0.0000	0.0000	0.0000	26.0000	0.0000	0.0000	0.0000	0.00000
7	15.0000	0.0000	14.0000	0.0000	27.0000	0.0000	2.4250	5.8023	0.45884
8	11.0338	9.1752	12.0000	0.0000	26.0338	9.1752	3.6614	12.7136	0.69051
9	8.9952	11.3179	15.0000	0.0000	20.0291	20.3020	9.1501	28.2548	0.10835
10	8.9290	9.9445	11.0338	9.1752	17.9243	21.6108	11.1066	31.9585	0.04559

Obs	Backlog Var	Shortfall Mean	Shortfall Var	Ready Rate	Fill Rate	Backorder Ratio	_STATUS_
1	1.96821	0.00000	0.00000	0.61529	0.91377	0.086231	SUCCESSFUL
2	0.60962	0.45884	1.14711	0.90758	0.98312	0.016883	SUCCESSFUL
3	0.34849	0.69051	2.54562	0.94819	0.98975	0.010245	SUCCESSFUL
4	0.60078	0.10835	0.42814	0.93916	0.98712	0.012882	SUCCESSFUL
5	0.23728	0.04559	0.17950	0.95913	0.99168	0.008325	SUCCESSFUL
6	0.00000	0.00000	0.00000	.	.	.	SUCCESSFUL
7	1.14711	0.00000	0.00000	0.73840	0.96187	0.038129	SUCCESSFUL
8	2.54562	0.00000	0.00000	0.74064	0.93720	0.062800	SUCCESSFUL
9	0.42814	0.00000	0.00000	0.95332	0.98787	0.012135	SUCCESSFUL
10	0.17950	0.00000	0.00000	0.97841	0.99494	0.005057	SUCCESSFUL

Example 3.17: Inventory Distribution for a Promotional or Seasonal Sale

This example demonstrates how to use the SYSTEM=PUSH option to distribute excess warehouse inventory to retailers for the purpose of a promotional or seasonal sale. It also demonstrates how to use the CAPACITY= option to prevent a retailer from receiving more inventory than it has the capacity to hold.

In the following inventorydata_push data set, each of three retailers, $R1$, $R2$, and $R3$, has zero on-hand inventory. The warehouse, W, has 80 units of inventory.

```
/* Inventory data set for 3-retailer serial network */
data inventorydata_push;
   input networkid $ skuloc $ amount;
   datalines;
N1 W  80
N1 R1 0
N1 R2 0
N1 R3 0
;
```

The following arcdata_push data set defines the network, which consists of a single warehouse that supplies the three retailers:

```
/* Arc data set for 3-retailer serial network */
data arcdata_push;
   input networkid $ predecessor $ successor $;
   datalines;
N1 external W
N1 W R1
N1 W R2
N1 W R3
;
```

In the following demanddata_push data set, each retailer is shown to have small demand requirements relative to the current on-hand inventory of 80 units at the warehouse, W. The retailer demands are easily satisfied by the on-hand inventory at the warehouse. However, for a promotional or seasonal sale, which is often time-sensitive, keeping excess inventory at the warehouse is not desired. Therefore, PROC MIRP should generate orders to distribute all excess warehouse inventory to the retailers in the most cost-effective way.

```
/* Demand data set for 3-retailer serial network */
data demanddata_push;
   input networkid $ skuloc $ period mean variance;
   datalines;
N1 R1 1 11 1
N1 R1 2 0 0
N1 R1 3 0 0
N1 R1 4 0 0
N1 R1 5 0 0
N1 R2 1 10 1
N1 R2 2 0 0
N1 R2 3 0 0
N1 R2 4 0 0
N1 R2 5 0 0
```

```
N1 R3 1 12 1
N1 R3 2 0 0
N1 R3 3 0 0
N1 R3 4 0 0
N1 R3 5 0 0
;
```

In the following nodedata_push data set, each retailer is shown to have a required service level of 90%, which is used to determine optimal policy parameters prior to generating orders. Whereas the expected demand at locations $R1$, $R2$, and $R3$ is 11, 10, and 12, respectively, the maximum inventory capable of being stored at any of these three locations is 35, as specified in the capacity variable in the nodedata_push data set.

```
/* Node data set for 3-retailer serial network */
data nodedata_push;
   input networkid $ skuloc $ holdingcost leadtime servicelevel capacity;
   datalines;
N1 W  1 1 0.9 .
N1 R1 1 1 0.9 35
N1 R2 1 1 0.9 35
N1 R3 1 1 0.9 35
;
```

The following call to PROC MIRP performs policy optimization and order generation by using the SYSTEM=PUSH option to distribute excess warehouse inventory to retailers. The resulting policy parameters and allocated quantity for each SKU-location are shown in Output 3.17.1.

```
title2 'Order Generation for a Promotional or Seasonal Sale';
proc mirp nodedata=nodedata_push arcdata=arcdata_push
          demanddata=demanddata_push inventorydata=inventorydata_push
 out=policy_order_push objective=policy_order system=push
 horizon=5;
run;
```

Output 3.17.1 Inventory Distribution for a Promotional or Seasonal Sale: SYSTEM=PUSH option

Two-Echelon Serial Network
Order Generation for a Promotional or Seasonal Sale

Obs	Network ID	Sku Loc	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock
1	N1	R1	1	1	12	13	2.00000
2	N1	R1	1	2	-1	0	0.00000
3	N1	R1	1	3	-1	0	0.00000
4	N1	R1	1	4	-1	0	0.00000
5	N1	R1	1	5	-1	0	0.00000
6	N1	R2	1	1	11	12	2.00000
7	N1	R2	1	2	-1	0	0.00000
8	N1	R2	1	3	-1	0	0.00000
9	N1	R2	1	4	-1	0	0.00000
10	N1	R2	1	5	-1	0	0.00000
11	N1	R3	1	1	13	14	2.00000
12	N1	R3	1	2	-1	0	0.00000
13	N1	R3	1	3	-1	0	0.00000
14	N1	R3	1	4	-1	0	0.00000
15	N1	R3	1	5	-1	0	0.00000
16	N1	W	2	1	40	41	1.99397
17	N1	W	2	2	0	1	0.99397
18	N1	W	2	3	-1	0	0.00000
19	N1	W	2	4	-1	0	0.00000
20	N1	W	2	5	-1	0	0.00000

Obs	Order Quantity	Allocated Quantity	Receipt Quantity	_STATUS_
1	13	35	0	SUCCESSFUL
2	.	.	35	SUCCESSFUL
3	.	.	0	SUCCESSFUL
4	.	.	0	SUCCESSFUL
5	.	.	0	SUCCESSFUL
6	12	18	0	SUCCESSFUL
7	.	.	18	SUCCESSFUL
8	.	.	0	SUCCESSFUL
9	.	.	0	SUCCESSFUL
10	.	.	0	SUCCESSFUL
11	14	27	0	SUCCESSFUL
12	.	.	27	SUCCESSFUL
13	.	.	0	SUCCESSFUL
14	.	.	0	SUCCESSFUL
15	.	.	0	SUCCESSFUL
16	0	0	80	SUCCESSFUL
17	.	.	0	SUCCESSFUL
18	.	.	0	SUCCESSFUL
19	.	.	0	SUCCESSFUL
20	.	.	0	SUCCESSFUL

Example 3.18: Policy Optimization with Starting Inventory

This example demonstrates how to include starting inventory in policy optimization and how the inventory affects the optimal policy in a multi-echelon network.

If you know the starting inventory at each location in your supply chain network, it is important to include this information in the policy optimization, especially when your network has multiple echelons. This is because the starting inventory at a downstream location determines its replenishment requirement, which in turn determines the optimal inventory levels at upstream locations.

The following inventorydata2_serial data set is created from the inventorydata_serial data set, with only one change: the on-hand inventory in period 1 is 40 units instead of 10. Clearly *R* has more inventory than it needs. PROC MIRP is called to optimize policy parameters based on the inventory status.

```
title2 'starting inventory for policy optimization';
data inventorydata2_serial;
   set inventorydata_serial;
   keep networkid skuloc period amount;
   if skuloc='R' and period=1 then amount=40;
run;

title2 'Policy Optimization with Starting Inventory';
proc mirp nodedata=nodedata_serial arcdata=arcdata_serial
          demanddata=demanddata_serial inventorydata=inventorydata2_serial
          out=optpolicy2_serial objective=optpolicy
           replications=5000 horizon=5;
run;
```

Output 3.18.1 shows that the policy parameters of *R* remain the same as those in Output 3.14.1, but the policy parameters at *W* are lower. In particular, the parameters in periods 1 and 2 are significantly lower. Because *R* has too much inventory in period 1, it does not order in period 1 and orders very little in period 2. Therefore, *W* does not need a lot of inventory for the first two periods.

Output 3.18.1 Policy Optimization with Starting Inventory: Two-Echelon Serial Network

Two-Echelon Serial Network
Policy Optimization with Starting Inventory

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	SERIAL	R	RETAILER 1	1	1	27	28	8.0000	SUCCESSFUL
2	SERIAL	R	RETAILER 1	1	2	28	29	8.0000	SUCCESSFUL
3	SERIAL	R	RETAILER 1	1	3	28	29	8.0000	SUCCESSFUL
4	SERIAL	R	RETAILER 1	1	4	27	28	8.0000	SUCCESSFUL
5	SERIAL	R	RETAILER 1	1	5	25	26	8.0000	SUCCESSFUL
6	SERIAL	W	WAREHOUSE	2	1	6	7	6.9005	SUCCESSFUL
7	SERIAL	W	WAREHOUSE	2	2	12	13	9.6971	SUCCESSFUL
8	SERIAL	W	WAREHOUSE	2	3	23	24	12.9063	SUCCESSFUL
9	SERIAL	W	WAREHOUSE	2	4	29	30	13.5088	SUCCESSFUL
10	SERIAL	W	WAREHOUSE	2	5	34	35	11.6279	SUCCESSFUL

Example 3.19: Bullwhip Effect in a Serial Network

This example illustrates how extremely important it is to account for the bullwhip effect when you manage inventory in a multi-echelon network. This example also demonstrates the importance of optimizing inventory in a multi-echelon network by using a network approach instead of a silo approach.

As discussed in the preceding single-location examples, the min-max policy or order constraints (such as minimum-order constraints or batch-size constraints) can produce the bullwhip effect. The following DATA step adds a batch-size constraint to location *R* in the serial network. The policy parameters are optimized again based on the constraint.

```
/* Add batch-size constraint to the retailer location */;
data nodedata_serial_batchsize;
   set nodedata_serial;
   if skuloc='R' then batchsize=20;
   else batchsize=.;
run;

title2 'Policy Optimization';
title3 'Batch-Size Constraint';
proc mirp nodedata=nodedata_serial_batchsize arcdata=arcdata_serial
          demanddata=demanddata_serial out=optpolicy_serial_batchsize
          objective=optpolicy replications=5000 horizon=5;
run;
```

As shown in Output 3.19.1, the reorder and order-up-to levels at *R* are much lower than those in Output 3.14.1. As discussed in Example 3.5, this is expected. The reorder and order-up-to levels at *W* are not much different from those in Output 3.14.1.

Output 3.19.1 Policy Optimization with Batch-Size Constraint: Two-Echelon Serial Network

Two-Echelon Serial Network
Policy Optimization
Batch-Size Constraint

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	SERIAL	R	RETAILER 1	1	1	22	23	3.000	SUCCESSFUL
2	SERIAL	R	RETAILER 1	1	2	23	24	3.000	SUCCESSFUL
3	SERIAL	R	RETAILER 1	1	3	23	24	3.000	SUCCESSFUL
4	SERIAL	R	RETAILER 1	1	4	22	23	3.000	SUCCESSFUL
5	SERIAL	R	RETAILER 1	1	5	20	21	3.000	SUCCESSFUL
6	SERIAL	W	WAREHOUSE	2	1	39	40	7.996	SUCCESSFUL
7	SERIAL	W	WAREHOUSE	2	2	39	40	10.100	SUCCESSFUL
8	SERIAL	W	WAREHOUSE	2	3	39	40	10.980	SUCCESSFUL
9	SERIAL	W	WAREHOUSE	2	4	39	40	13.084	SUCCESSFUL
10	SERIAL	W	WAREHOUSE	2	5	59	60	32.904	SUCCESSFUL

You might find the results at *W* surprising because the batch-size constraint produces the bullwhip effect: *W* experiences much higher demand variance from *R* than the variance of customer demand that *R* experiences. If so, you would expect that much more inventory is needed at *W* to account for the increased variance. But results do not appear to be that way. To verify the correctness of the results, you can use the policy parameters in the optpolicy_serial_batchsize data set and run the MIRP procedure as follows to estimate KPIs:

```
title2 'KPI Prediction';
title3 'Batch-Size Constraint';
proc mirp nodedata=nodedata_serial_batchsize arcdata=arcdata_serial
          demanddata=demanddata_serial inventorydata=optpolicy_serial_batchsize
          out=predictkpi_serial_batchsize objective=predictkpi
          replications=5000 horizon=5;
run;
```

Output 3.19.2 KPI Prediction with Batch-Size Constraint: Two-Echelon Serial Network

Two-Echelon Serial Network
KPI Prediction
Batch-Size Constraint

Obs	Network ID	Sku Loc	Description	Echelon	Period	External Demand Mean	External Demand Var	Internal Demand Mean	Internal Demand Var
1	SERIAL	R	RETAILER 1	1	1	9	9	0.000	0.000
2	SERIAL	R	RETAILER 1	1	2	11	11	0.000	0.000
3	SERIAL	R	RETAILER 1	1	3	10	10	0.000	0.000
4	SERIAL	R	RETAILER 1	1	4	11	11	0.000	0.000
5	SERIAL	R	RETAILER 1	1	5	9	9	0.000	0.000
6	SERIAL	W	WAREHOUSE	2	1	0	0	11.016	99.148
7	SERIAL	W	WAREHOUSE	2	2	0	0	9.912	100.012
8	SERIAL	W	WAREHOUSE	2	3	0	0	11.076	99.182
9	SERIAL	W	WAREHOUSE	2	4	0	0	8.912	98.836
10	SERIAL	W	WAREHOUSE	2	5	0	0	9.032	99.243

Obs	Order Mean	Order Var	Planned Receipt Mean	Planned Receipt Var	Pipeline Mean	Pipeline Var	OnHand Mean	OnHand Var	Backlog Mean
1	11.016	99.148	9.0366	99.105	11.016	99.148	12.0285	48.150	0.08035
2	9.912	100.012	11.0160	99.148	9.952	100.018	12.0860	50.183	0.11742
3	11.076	99.182	9.9520	100.018	11.076	99.182	12.1120	52.485	0.12151
4	8.912	98.836	11.0760	99.182	8.868	98.738	12.1618	51.102	0.10780
5	9.032	99.243	8.8680	98.738	9.036	99.091	12.0816	51.344	0.13466
6	0.004	0.080	11.0120	99.476	28.904	98.979	11.0920	105.069	0.04000
7	1.884	34.137	28.9000	98.970	1.888	34.202	30.0400	105.780	0.00000
8	8.148	96.589	0.0040	0.080	10.032	102.740	18.9680	100.075	0.00000
9	11.004	99.332	1.8840	34.137	19.152	81.057	11.9840	109.206	0.04400
10	28.908	98.827	8.1480	96.589	39.912	77.768	11.0960	105.860	0.04000

Obs	Backlog Var	Shortfall Mean	Shortfall Var	Ready Rate	Fill Rate	Backorder Ratio	_STATUS_
1	0.29226	0.040	0.79856	0.96368	0.99111	0.008894	SUCCESSFUL
2	0.49203	0.000	0.00000	0.95039	0.98932	0.010679	SUCCESSFUL
3	0.45714	0.000	0.00000	0.94659	0.98776	0.012236	SUCCESSFUL
4	0.44014	0.044	0.87824	0.95397	0.99021	0.009788	SUCCESSFUL
5	0.52655	0.040	0.79856	0.94430	0.98500	0.015003	SUCCESSFUL
6	0.79856	0.000	0.00000	0.99637	0.99637	0.003631	SUCCESSFUL
7	0.00000	0.000	0.00000	1.00000	1.00000	0.000000	SUCCESSFUL
8	0.00000	0.000	0.00000	1.00000	1.00000	0.000000	SUCCESSFUL
9	0.87824	0.000	0.00000	0.99506	0.99506	0.004937	SUCCESSFUL
10	0.79856	0.000	0.00000	0.99557	0.99557	0.004429	SUCCESSFUL

Output 3.19.2 shows that the average ready rate at R is $(0.96368 + 0.95039 + 0.94519 + 0.94997 + 0.9443)/5 = 0.950706$ over five periods, which is very close to the target service level of 95%.

At *W*, the average internal demand is about 10, which is close to the average demand at *R*. However, the variance of the internal demand is close to 100, which is much higher than the variance of the demand at *R*. Clearly this is the bullwhip effect.

The average ready rate at *W* is $(0.99637 + 0.99314 + 0.98591 + 0.99282 + 0.99557)/5 = 0.99276$, which is higher than the target level of 95%. You might find this condition quite often, because the order stream from *R* to *W* is not as smooth as the customer demand; the orders arrive at *W* in batches of 20 as a result of the batch-size constraint. Because of the lumpiness in the order stream, it is very likely that you cannot hit the target service level as closely as you want. You might have to settle for a higher service level. This is acceptable as long as the service level at *R* is close to the target level.

The policy parameters in Output 3.19.1 are optimized jointly. In other words, the MIRP procedure takes into account the interactions of the two locations in the network when it optimizes their policy parameters. This is called the *network* approach. You can also calculate policy parameters for *R* and *W* separately; that is, you can treat them as two independent locations. This is called the *silo* approach, and it is widely used in practice because of its simplicity. The following DATA steps create input data sets for PROC MIRP to optimize policy parameters at *W*:

```
data nodedata_serial_warehouse;
   set nodedata_serial;
   if skuloc='W';
run;

data arcdata_serial_warehouse;
   set arcdata_serial;
   if successor='W';
run;
```

In the following DATA step, the demand forecast in the demanddata_serial_warehouse data set is set to the internal demand from the predictkpi_serial_batchsize data set. This is consistent with the practice in which the demand at an internal location is forecast based on the orders that it receives from downstream locations.

```
data demanddata_serial_warehouse;
   set predictkpi_serial_batchsize;
   keep networkid skuloc period internaldemandmean internaldemandvar;
   if skuloc='W';
run;
```

The following call to PROC MIRP calculates policy parameters at *W*. The MAXCV=1.5 option accounts for the fact that the coefficient of variation of the forecast at *W* ranges from 0.9 to 1.12. If you do not add the MAXCV= option, the default value is 1. The MIRP procedure trims the variance if it finds the coefficient of variation in a period is greater than the value of the MAXCV= option.

```
title2 'Policy Optimization';
title3 'The warehouse is optimized separately from the retailer location';
proc mirp nodedata=nodedata_serial_warehouse arcdata=arcdata_serial_warehouse
          demanddata=demanddata_serial_warehouse out=optpolicy_serial_warehouse
          objective=optpolicy replications=5000 horizon=5 MAXCV=1.5;
   demand/mean=internaldemandmean var=internaldemandvar;
run;
```

Output 3.19.3 shows that the order-up-to levels are significantly higher than those in Output 3.19.1. You know you can achieve a 95% ready rate with the policy parameters in Output 3.19.1. Therefore, the results in Output 3.19.3 indicate that you are keeping too much inventory if you calculate policy parameters at *W* independently from *R*. This is the problem with the silo optimization. Because policy parameters are calculated independently, the approach cannot capture the interactions between locations within the same network. The bullwhip effect in this example does not reflect the true variation of customer demand, but it is treated as the variation of customer demand in the silo optimization.

Output 3.19.3 Policy Optimization at the Warehouse Separate from the Retailer Location

Two-Echelon Serial Network
Policy Optimization
The warehouse is optimized separately from the retailer location

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	SERIAL	W	WAREHOUSE	1	1	65	66	33.996	SUCCESSFUL
2	SERIAL	W	WAREHOUSE	1	2	64	65	35.100	SUCCESSFUL
3	SERIAL	W	WAREHOUSE	1	3	64	65	35.980	SUCCESSFUL
4	SERIAL	W	WAREHOUSE	1	4	63	64	37.024	SUCCESSFUL
5	SERIAL	W	WAREHOUSE	1	5	63	64	36.904	SUCCESSFUL

To summarize, the silo optimization is simple. But because it fails to capture the bullwhip effect correctly, it might suggest inventory levels that are much higher than needed. The network optimization, which PROC MIRP uses, can capture the bullwhip effect and therefore produces optimal policy parameters.

Example 3.20: Evaluation of Internal Service Level

This example demonstrates how to use the OBJECTIVE=EVALISL option to evaluate the average cost over a long term for a particular service-level setting. It also shows how the service-level setting at an internal location affects the cost of an entire network.

When you optimize policy parameters for a multi-echelon network, you need to specify service-level targets for every location in the network. You can also use the MIRP procedure to determine average inventory costs based on your service-level setting. Because service-level targets are normally mid-term or long-term goals, the average inventory costs are estimated using average demand forecast over a relatively long planning horizon.

In Example 3.14, the demand forecast varies slightly from periods 2 through 5. The average demand is about 10, and the average variance is also around 10. The following DATA step creates the demanddata_serial data set with a stationary demand forecast that has a mean of 10 and a variance of 10:

```
/* Demand data set for two-echelon serial network */
data demanddata_serial;
    infile datalines dlm=',';
    input   networkid: $6.
            skuloc: $2.
            mean
            variance;
datalines;
SERIAL, R, 10, 10
;
```

Now you can use the newly created data set together with the existing nodedata_serial and arcdata_serial data sets to project the average inventory cost for the current service-level setting (that is, a ready rate of 95% for both locations in the network). In the following statements, the OBJECTIVE=EVALISL option requests that PROC MIRP evaluate a set of performance indicators over a long run. The POLICYPARM=DOUBLE option requests that the procedure produce noninteger policy parameters, and the DEMANDMODEL=CONTINUOUS option specifies that a continuous distribution be used. Because you are interested in the long-run average, it is recommended that you set these two options in this way.

```
title2 'Internal Service Level Evaluation';
proc mirp nodedata=nodedata_serial arcdata=arcdata_serial
          demanddata=demanddata_serial out=evalisl_serial OBJECTIVE=EVALISL
          POLICYPARM=DOUBLE DEMANDMODEL=CONTINUOUS;
run;
```

Output 3.20.1 shows the order-up-to levels at each location. These levels are averages over a long run based on the stationary demand in the demanddata_serial data set. They are fairly close to the levels in Output 3.14.1. You can also find on-hand inventory and backlog amount at each location. For example, the average ending on-hand inventory is 7.5383 at R and 9.14088 at W. The on-hand holding cost is calculated as follows:

- At R, the unit holding cost is 7, so the on-hand holding cost is $7 \times 7.5383 = 52.7681$.
- At W, the unit holding cost is 6.8, so the on-hand holding cost is $6.8 \times 9.14088 = 62.157984$.

If you have the unit cost of the inventory, you can also calculate the inventory cost at each location based on the estimates of the on-hand inventory.

Output 3.20.1 Service-Level Evaluation: Two-Echelon Serial Network

Two-Echelon Serial Network
Internal Service Level Evaluation

Obs	Network ID	Sku Loc	Description	Ready Rate	Fill Rate	Backorder Ratio	Echelon	Order UpTo Level	External Demand Mean
1	SERIAL	R	RETAILER 1	0.95	0.99063	0.009368	1	27.5576	10
2	SERIAL	W	WAREHOUSE	0.95	0.98865	0.011353	2	39.0273	0

Obs	External Demand Var	Internal Demand Mean	Internal Demand Var	Pipeline Mean	Pipeline Var	OnHand Mean	OnHand Var	Backlog Mean	Backlog Var
1	10	0	0	10	10	7.53830	18.7217	0.09422	0.30985
2	0	10	10	20	20	9.14088	27.4724	0.11353	0.45201

Obs	Shortfall Mean	Shortfall Var	Delay Mean	DelayVar	Safety Stock	Oh Holding Cost	_STATUS_
1	0.11353	0.45201	0.009422	.0021564	7.55762	52.7681	SUCCESSFUL
2	0.00000	0.00000	0.011353	.0033847	9.02734	62.1580	SUCCESSFUL

So far you have evaluated the average cost for a given set of service-level targets. You can also evaluate the cost over a range of service-level targets. The following macro repeatedly calls the MIRP procedure with ready rate targets at *W* that range from 10% to 99% and then stores the total network cost in the ReadyRate_Impact data set.

```
%macro ReadyRate_Impact_EVALISL();
   options nonotes;
      %let rr=0.1;
      %do %while(%sysevalf(&rr<0.99));
         proc sql;
            update nodedata_serial
               set servicelevel=&rr where skuloc='W';
         quit;

         proc mirp nodedata=nodedata_serial arcdata=arcdata_serial
                   demanddata=demanddata_serial OUT=evalisl_serial
                   objective=EVALISL policyparm=double;
         run;
```

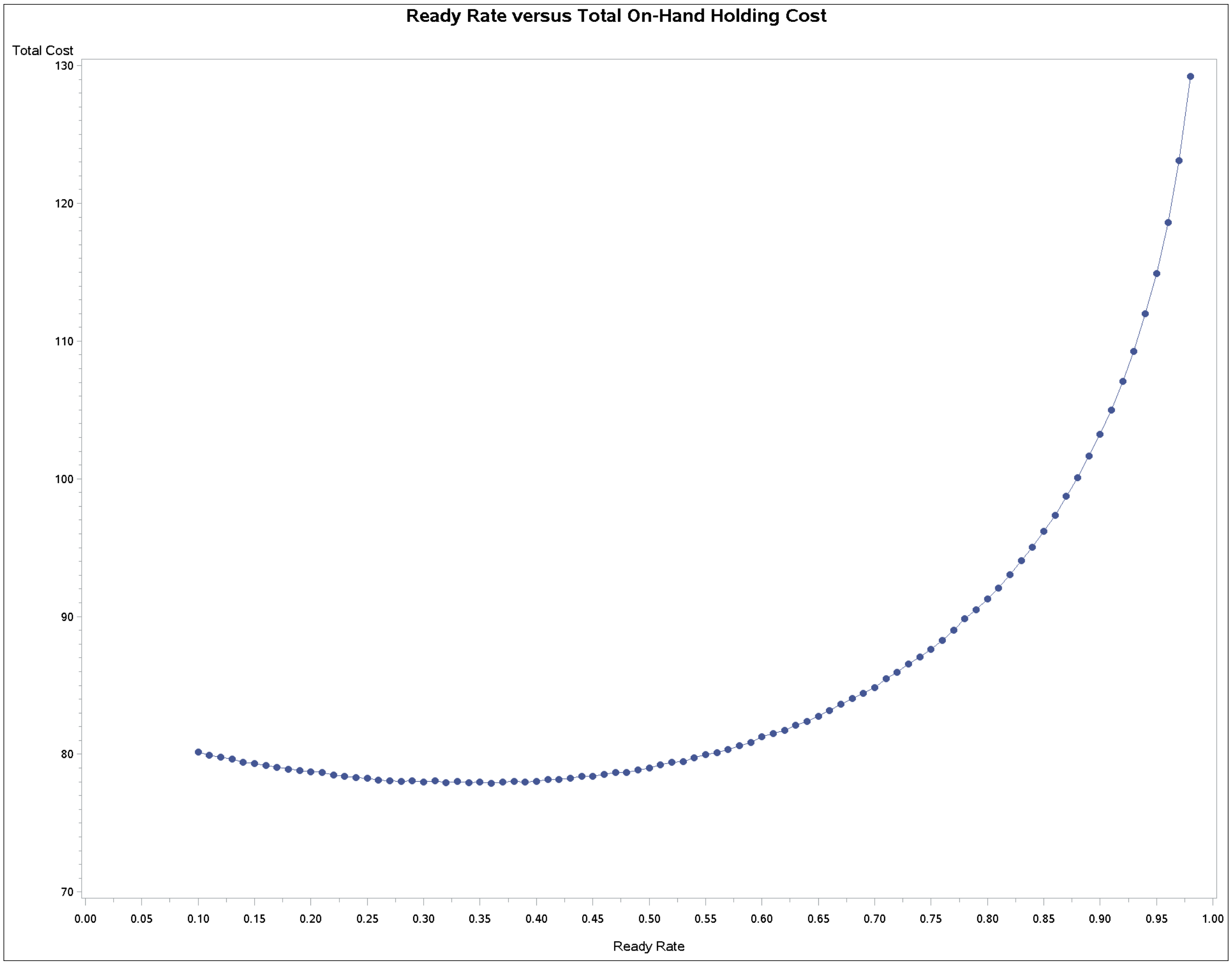
```
         proc sql;
            create table tmp_rr as
               select &rr as readyrate label='Ready Rate',
                          sum(ohholdingcost) as totalcost label='Total Cost'
                          from evalisl_serial;
         quit;

         %if %sysevalf(&rr<0.101) %then %do;
            data ReadyRate_Impact;
               set tmp_rr;
            run;
         %end;
         %else %do;
            proc append base=ReadyRate_Impact data=tmp_rr;run;
         %end;

         %let rr=%sysevalf(&rr+0.01);
      %end;
   options notes;
%mend;
%ReadyRate_Impact_EVALISL;
```

You can use the GPLOT procedure as follows to plot the total network cost against the ready rate at *W*:

```
goptions reset=all border;
title1 "Ready Rate versus Total On-Hand Holding Cost";
  symbol1 interpol=join value=dot;
  proc gplot data=ReadyRate_impact;
   plot totalcost*readyrate / haxis=0 to 1 by 0.05;
run;
quit;
```

Output 3.20.2 shows a fairly smooth curve whose minimum is around 0.35, indicating that the total network cost can be minimized if the ready rate at *W* is set around 35%. If you set the ready rate at 80%, the total network cost increases by about 10%. With the current setting of 95%, the cost is more than 45% higher than the minimum. Clearly, if cost is the only concern, you need to lower the ready rate to 35%. However, if cost is not the only concern, this curve helps you make a proper trade-off between inventory costs and other factors.

Output 3.20.2 Service-Level Impact on Cost: Two-Echelon Serial Network



Example 3.21: Optimization of Internal Service Level

This example demonstrates how to determine the optimal service level for an internal location of a multi-echelon network. An internal location is a location that does not directly face customer demand. The OBJECTIVE=OPTISL option is used.

Example 3.20 plots the network cost against the ready rate at *W*. From the curve in Output 3.20.2, you can see where the minimum lies. If you want to know where the minimum is without creating the plot, you can do that by using the OBJECTIVE=OPTISL option as follows:

```
title2 'Internal Service Level Optimization';
proc mirp nodedata=nodedata_serial arcdata=arcdata_serial
          demanddata=demanddata_serial  OUT=optisl_serial objective=optisl
          policyparm=double demandmodel=continuous;
run;
```

Output 3.21.1 shows that the optimal ready rate at *W* is 33.661% and the minimum network cost is $69.6011 + 8.3302 = 77.9313$. These values are consistent with what is observed in Example 3.20.

Output 3.21.1 Service-Level Optimization: Two-Echelon Serial Network

Ready Rate versus Total On-Hand Holding Cost
Internal Service Level Optimization

Obs	Network ID	Sku Loc	Description	Ready Rate	Fill Rate	Backorder Ratio	Echelon	Order UpTo Level	External Demand Mean
1	SERIAL	R	RETAILER 1	0.95000	0.98945	0.01055	1	33.3438	10
2	SERIAL	W	WAREHOUSE	0.33661	0.65527	0.35248	2	27.7002	0

Obs	External Demand Var	Internal Demand Mean	Internal Demand Var	Pipeline Mean	Pipeline Var	OnHand Mean	OnHand Var	Backlog Mean	Backlog Var
1	10	0	0	10	10	9.94302	32.5573	0.12411	0.5382
2	0	10	10	20	20	1.22503	5.8005	3.52484	15.5634

Obs	Shortfall Mean	Shortfall Var	Delay Mean	Delay Var	Safety Stock	Oh Holding Cost	_STATUS_
1	3.52484	15.5634	0.01241	0.00414	13.3438	69.6011	SUCCESSFUL
2	0.00000	0.0000	0.35248	0.12039	-2.2998	8.3302	SUCCESSFUL

Example 3.22: Policy Optimization in a Distribution Network

This example demonstrates the use of the MIRP procedure in the policy optimization for a two-echelon distribution network. It also shows a short version of the DEMANDDATA= data set when the demand forecast is stationary. The safety stock calculation under the min-max policy is explained.

The following DATA steps create three input data sets. The distribution network has one warehouse and two retailer locations. Service levels are set at a 95% ready rate for all locations. The min-max policy is used at the retailer locations, producing the bullwhip effect at the warehouse. As discussed in Example 3.19, it is important to solve all three locations simultaneously in order to capture the bullwhip effect correctly without overestimating the inventory requirement at the warehouse.

```
title 'Two-Echelon Distribution Network';
/* Node data set for two-echelon distribution network */
data nodedata_dist;
    infile datalines dlm=',';
    input   networkid:$4.
            skuloc: $2.
            description: $16.
            leadTime
            holdingCost
            serviceLevel
            serviceType: $2.
            policy: $2.
            pbr;
datalines;
```

```
DIST, W,  WAREHOUSE, 2, 4, 0.95, RR, BS, 1
DIST, R1, RETAILER1, 1, 7, 0.95, RR, SS, 2
DIST, R2, RETAILER2, 1, 7, 0.95, RR, SS, 2
;


/* Arc data set for two-echelon distribution network */
data arcdata_dist;
    infile datalines dlm=',';
    input   networkid: $4.
            predecessor: $8.
            successor: $2.;
datalines;
DIST, EXTERNAL, W
DIST, W, R1
DIST, W, R2
;
```

There is no PERIOD variable in the following demanddata_dist data set. In this case, the MIRP procedure assumes stationary demand (that is, it assumes that the demand forecast is the same in every period in the planning horizon).

```
/* Demand data set for two-echelon distribution network */
data demanddata_dist;
    infile datalines dlm=',';
    input   networkid: $4.
            skuloc: $2.
            mean
            variance;
datalines;
DIST, R1, 10, 25
DIST, R2, 80, 1000
;
```

The following call to PROC MIRP does not include the OBJECTIVE= option. By default, OBJECTIVE=OPTPOLICY. So the procedure optimizes policy parameters for the distribution network.

```
title2 'Policy Optimization';
proc mirp nodedata=nodedata_dist arcdata=arcdata_dist
          demanddata=demanddata_dist out=optpolicy_dist
          horizon=5 replications=5000;
run;
```

Output 3.22.1 shows a stationary policy for all locations in the network because the demand forecast is stationary. When the min-max policy is used, the safety stock is calculated as (reorder level − average demand during (lead time + 1) periods). For example, at $R1$ the reorder level is 26, the average demand is 10 per period, and the lead time is 1. The safety stock at $R1$ is then $(26 - 10 \times 2) = 6$.

Output 3.22.1 Policy Optimization: Two-Echelon Distribution Network

Two-Echelon Distribution Network
Policy Optimization

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	DIST	R1	RETAILER1	1	1	26	41	21.000	SUCCESSFUL
2	DIST	R1	RETAILER1	1	2	26	41	21.000	SUCCESSFUL
3	DIST	R1	RETAILER1	1	3	26	41	21.000	SUCCESSFUL
4	DIST	R1	RETAILER1	1	4	26	41	21.000	SUCCESSFUL
5	DIST	R1	RETAILER1	1	5	26	41	21.000	SUCCESSFUL
6	DIST	R2	RETAILER2	1	1	189	304	144.000	SUCCESSFUL
7	DIST	R2	RETAILER2	1	2	189	304	144.000	SUCCESSFUL
8	DIST	R2	RETAILER2	1	3	189	304	144.000	SUCCESSFUL
9	DIST	R2	RETAILER2	1	4	189	304	144.000	SUCCESSFUL
10	DIST	R2	RETAILER2	1	5	189	304	144.000	SUCCESSFUL
11	DIST	W	WAREHOUSE	2	1	407	408	137.785	SUCCESSFUL
12	DIST	W	WAREHOUSE	2	2	408	409	137.202	SUCCESSFUL
13	DIST	W	WAREHOUSE	2	3	408	409	137.084	SUCCESSFUL
14	DIST	W	WAREHOUSE	2	4	408	409	137.153	SUCCESSFUL
15	DIST	W	WAREHOUSE	2	5	551	552	281.699	SUCCESSFUL

Example 3.23: Order Generation in a Distribution Network

This example demonstrates how the MIRP procedure allocates inventory at an upstream location when it cannot satisfy all orders from downstream locations.

The following inventorydata_dist data set is derived from the output data set optpolicy_dist from Example 3.22. The on-hand and in-transit inventory are added.

```
/* Inventory data set for two-echelon distribution network */
data inventorydata_dist;
   set optpolicy_dist;
   if skuloc='R1' then do;
         if period=1 then amount=18;
         else amount=.;
      end;
   else if skuloc='R2' then do;
         if period=1 then amount=100;
         else amount=.;
      end;
   else do;
         if period=1 then amount=200;
         else if period=2 then amount=80;
         else amount=.;
         end;
run;
```

The current total inventory at $R1$ is 18 units, which is below the reorder level of 26 units. A replenishment order of $41 - 18 = 23$ units is suggested. The current total inventory at $R2$ is 100 units, which is below the reorder level of 189 units. A replenishment order of $304 - 100 = 204$ units is suggested. Therefore, the total order quantity that is received at W is $23 + 204 = 227$ units, which is more than the on-hand inventory of 200 units at W. The following call to PROC MIRP generates order quantities for the locations in the distribution network:

```
title2 'Order Generation';
proc mirp nodedata=nodedata_dist arcdata=arcdata_dist
          demanddata=demanddata_dist
          inventorydata=inventorydata_dist out=createorder_dist
          objective=createorder horizon=5 replications=5000;
run;
```

Output 3.23.1 shows that the suggested order quantities are 23 units and 204 units for $R1$ and $R2$, respectively. Because W does not have sufficient inventory to satisfy all orders, the MIRP procedure determines the allocation of the available inventory based on demand volume, service-level targets, and current inventory status at each downstream location. In this example, PROC MIRP allocates 19 units to $R1$ and 181 units to $R2$.

Output 3.23.1 Order Generation: Two-Echelon Distribution Network

Two-Echelon Distribution Network
Order Generation

Obs	Network ID	Sku Loc	Description	Echelon	Period	Order Quantity	Allocated Quantity	_STATUS_
1	DIST	R1	RETAILER1	1	1	23	19	SUCCESSFUL
2	DIST	R2	RETAILER2	1	1	204	181	SUCCESSFUL
3	DIST	W	WAREHOUSE	2	1	128	128	SUCCESSFUL

Example 3.24: Impact of Customer Delivery Time

This example demonstrates how customer delivery time affects inventory costs in an assembly network.

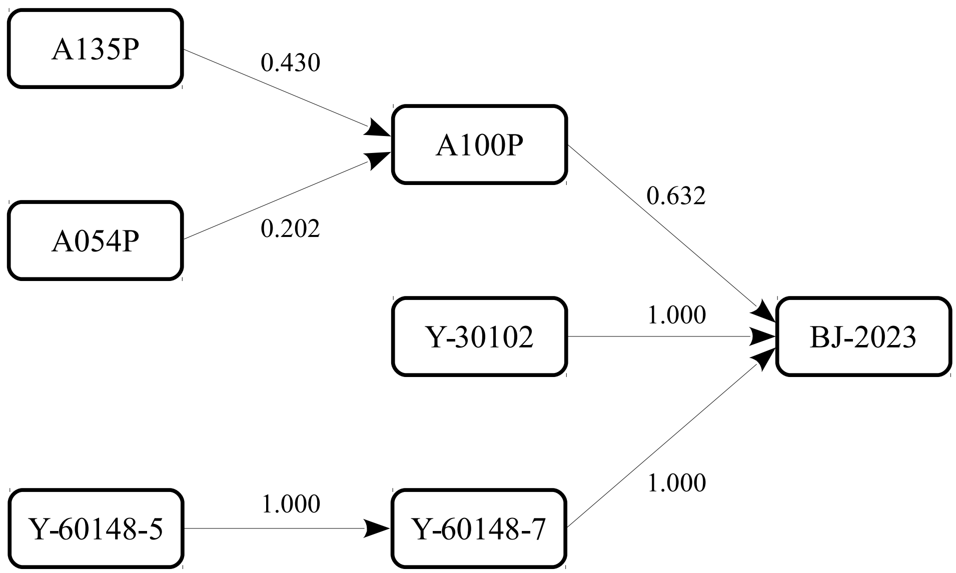
In many industries, customers' demand must be satisfied immediately. This is especially true in the retail business. Customers usually do not want to wait a few days to get their groceries. However, in the manufacturing industry, advance orders are quite common. Customers place orders and are willing to wait for a period of time to have their orders fulfilled. This period of time is called customer delivery time.

In this example, a three-echelon assembly network is used to show how to take advantage of the customer delivery time to reduce inventory investment. There is one finished product, BJ-2023, with an average demand of 63 and a variance of 357, as defined in the following DATA step:

```
data demand_asm;
   format material $16. component $16.;
   input material $ component $ mean variance;
datalines;
BJ-2023  BJ-2023  63 357
;
```

The assembly network consists of four raw materials, A135P, A054P, Y-60148-5, and Y-30102, which are purchased from external suppliers. A135P and A054P are combined into A100P. Then Y-60148-5 is converted into Y-60148-7. Finally, A100P, Y-30102, and Y-60148-7 are assembled into BJ-2023. This assembly network, along with the bill of material (BOM) quantities, is illustrated in Figure 3.4. Some of the BOM quantities are not integers; this suggests that the process might be a chemical process.

Figure 3.4 Diagram of Three-Echelon Assembly Network



The assembly network in Figure 3.4 is represented by the arc_asm data set, as follows:

```
data arc_asm;
   format material $10. predecessor $16. successor $16.;
   input material $ predecessor $ successor $ quantity;
datalines;
BJ-2023  EXTERNAL    A135P       1.000
BJ-2023  EXTERNAL    A054P       1.000
BJ-2023  EXTERNAL    Y-60148-5   1.000
BJ-2023  EXTERNAL    Y-30102     1.000
BJ-2023  A135P       A100P       0.430
BJ-2023  A054P       A100P       0.202
BJ-2023  Y-60148-5   Y-60148-7   1.000
BJ-2023  A100P       BJ-2023     0.632
BJ-2023  Y-30102     BJ-2023     1.000
BJ-2023  Y-60148-7   BJ-2023     1.000
;
```

In the following node_asm data set, you can see fairly long lead times. For example, it takes 29 periods to make BJ-2023 from A100P, Y-30102, and Y-60148-7, and it takes 82 periods to receive delivery of Y-60148-5 from an external supplier.

```
data node_asm;
   format material $16. Component $16.;
   input material $ component $ servicelevel leadtime holdingcost;
   datalines;
BJ-2023  BJ-2023     0.95    29  3.2
BJ-2023  Y-60148-7   0.95    83  1.2
BJ-2023  Y-60148-5   0.95    82  1.0
BJ-2023  Y-30102     0.95    71  1.0
BJ-2023  A100P       0.95    1   0.7
BJ-2023  A054P       0.95    16  1.0
BJ-2023  A135P       0.95    16  1.0
;
```

The following call to PROC MIRP projects the total cost of the network. You use the NETWORKID= and SKULOC= options to specify the variable names because they are different from the default variable names. The cost projection is shown in Output 3.24.1.

```
title 'Three-Echelon Assembly Network';
title2 'Cost Summary';
proc mirp nodedata=node_asm arcdata=arc_asm demanddata=demand_asm
          out=out_evalisl (keep=networkid skuloc echelon ohholdingcost)
          objective=evalisl policyparm=double;
   node/networkid=material skuloc=component;
   arc/networkid=material;
   demand/networkid=material skuloc=component;
run;
```

Output 3.24.1 Cost Summary: Three-Echelon Assembly Network

Three-Echelon Assembly Network
Cost Summary

Obs	Network ID	SkuLoc	Echelon	Oh Holding Cost
1	BJ-2023	BJ-2023	1	554.460
2	BJ-2023	A100P	2	20.917
3	BJ-2023	Y-30102	2	269.980
4	BJ-2023	Y-60148-7	2	353.575
5	BJ-2023	A054P	3	16.596
6	BJ-2023	A135P	3	35.456
7	BJ-2023	Y-60148-5	3	290.371

If customers expect to have their orders fulfilled right away, you need to keep inventory of the finished products (BJ-2023). If customers are willing to place their orders 29 periods in advance, then you do not need to keep inventory of BJ-2023 because you have enough time to produce BJ-2023 from other components. If you do not keep inventory of BJ-2023, you reduce the inventory investment by 554.46 units, which is the expected on-hand holding cost of the finished products. If customers are willing to wait even longer, you can remove additional inventory from the network. Table 3.6 summarizes the cost impact.

Table 3.6 Cost Impact by Customer Delivery Time

Customer Delivery Time	SKU That Does Not Need Inventory	Cost Reduction	Total Inventory Holding Cost
0		0	1541.355
29	BJ-2023	554.460	986.895
30	A100P	20.917	965.978
46	A135P	35.456	930.522
46	A054P	16.596	913.926
100	Y-30102	269.980	643.946
112	Y-60148-7	353.575	290.371
194	Y-60148-5	290.371	0

Example 3.25: A Silo Solution versus a Network Solution

In a multi-echelon supply chain network, you can compute the inventory replenishment policies for each location independently from other locations in the network, resulting in what is referred to as a "silo solution." This approach is problematic, because it ignores the interactions between predecessor and successor locations in the network. The interactions are twofold:

1. Less than perfect service levels at upstream locations create backlogs that affect the replenishment at downstream locations. When the service level at a predecessor is less than 100%, fulfillment shortages could occur when a successor places a replenishment order. Therefore, the calculation of the inventory policies at the successor must take these shortages into account.

2. The inventory policies at downstream locations affect the inventory policies at upstream locations. The inventory policies at a successor determine its order quantities, which become the demand at a predecessor. The inventory policies at the predecessor are functions of its demand. Therefore, the inventory policies at the successor indirectly affect the inventory policies at the predecessor.

These interactions are important factors, and PROC MIRP takes them into account. In other words, PROC MIRP solves for all locations in a network jointly by using a *network solution*. Ignoring the interaction might lead to a service-level performance that is significantly lower than what is targeted. This problem is demonstrated by this example.

The supply chain network that is considered here is exactly the same as the one in the section "Two-Echelon Assembly Network" on page 63. To produce an independent solution, three new networks are created, each of which corresponds to a location in the original network. These new networks are described in the following data sets:

```
data nodedata6;
  infile datalines dlm=',';
  input networkid: $2.
        skuloc: $2.
        description: $15.
        lt sl hc;
datalines;
N4, AP, component 1,     1, 0.5,  1
N5, BP, component 2,     2, 0.5 , 2
N6, FP, finished goods,   3, 0.95, 13
;

data arcdata6;
  infile datalines dlm=',';
  input networkid: $3.
       head: $9.
        tail: $3.
        qty;
datalines;
N4, EXTERNAL, AP, 1
N5, EXTERNAL, BP, 1
N6, EXTERNAL, FP, 1
;

data demanddata6;
  infile datalines dlm=',';
  input networkid: $3.
        skuloc: $3.
        period mean variance;
datalines;
N4, AP, 1, 48, 324
N5, BP, 1, 32, 144
N6, FP, 1, 16, 36
;
```

Following are the differences between these data sets and those in the section "Two-Echelon Assembly Network" on page 63:

- The networkid variable assigns a unique value to each of the three locations.
- In the arcdata6 data set, all three locations are linked to an external supplier because they are treated independently, not as part of a network. Note that the BOM quantities are also changed accordingly.
- In the demanddata6 data set, demand at the two components is also provided. Their demands are equal to the demand from the finished goods multiplied by the BOM quantities. Such a demand population is valid only under a very strict condition: all downstream locations must use the base-stock policy and must have stationary demand. Such a condition rarely exists in real applications.

The following call to PROC MIRP computes the reorder and order-up-to levels for each location independently:

```
title 'A Silo Solution vs. a Network Solution';
title2 'Silo Solution';
proc mirp nodedata=nodedata6 arcdata=arcdata6 demanddata=demanddata6
          out=out_example6 horizon=8;
run;
```

Output 3.25.1 contains the policy parameters from the independent solution. Note that the parameters are quite different from those in the "Two-Echelon Assembly Network" on page 63.

Output 3.25.1 Output Data Set of the Silo Solution

A Silo Solution vs. a Network Solution
Silo Solution

Obs	Network ID	Sku Loc	Description	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	N4	AP	component 1	1	1	97	98	2	SUCCESSFUL
2	N4	AP	component 1	1	2	97	98	2	SUCCESSFUL
3	N4	AP	component 1	1	3	97	98	2	SUCCESSFUL
4	N4	AP	component 1	1	4	97	98	2	SUCCESSFUL
5	N4	AP	component 1	1	5	97	98	2	SUCCESSFUL
6	N4	AP	component 1	1	6	97	98	2	SUCCESSFUL
7	N4	AP	component 1	1	7	97	98	2	SUCCESSFUL
8	N4	AP	component 1	1	8	97	98	2	SUCCESSFUL
9	N5	BP	component 2	1	1	96	97	1	SUCCESSFUL
10	N5	BP	component 2	1	2	96	97	1	SUCCESSFUL
11	N5	BP	component 2	1	3	96	97	1	SUCCESSFUL
12	N5	BP	component 2	1	4	96	97	1	SUCCESSFUL
13	N5	BP	component 2	1	5	96	97	1	SUCCESSFUL
14	N5	BP	component 2	1	6	96	97	1	SUCCESSFUL
15	N5	BP	component 2	1	7	96	97	1	SUCCESSFUL
16	N5	BP	component 2	1	8	96	97	1	SUCCESSFUL
17	N6	FP	finished goods	1	1	84	85	21	SUCCESSFUL
18	N6	FP	finished goods	1	2	84	85	21	SUCCESSFUL
19	N6	FP	finished goods	1	3	84	85	21	SUCCESSFUL
20	N6	FP	finished goods	1	4	84	85	21	SUCCESSFUL
21	N6	FP	finished goods	1	5	84	85	21	SUCCESSFUL
22	N6	FP	finished goods	1	6	84	85	21	SUCCESSFUL
23	N6	FP	finished goods	1	7	84	85	21	SUCCESSFUL
24	N6	FP	finished goods	1	8	84	85	21	SUCCESSFUL

You can use PROC MIRP to evaluate the performance of a set of policy parameters. This functionality can be used here to see whether the independent solution indeed achieves the target service levels.

The following DATA step stores the optimized policy parameters from the previous call to PROC MIRP in a data set called inventorydata7. The NetworkID is changed to a constant value for all locations so that the policy obtained from treating the locations as independent networks can be evaluated with respect to the real setting of a single network.

```
data inventorydata7;
   set out_example6;
   keep NetworkID SkuLoc
        Period ReorderLevel OrderUpToLevel;
   NetworkID="N3";
```

The following call to PROC MIRP projects KPIs for the single network, based on the optimized policy parameters from the independent locations.

```
title 'An Independent Solution vs. a Joint Solution';
title2 'Evaluating Silo Solution';
proc mirp nodedata=nodedata3 arcdata=arcdata3
          demanddata=demanddata3
          inventorydata=inventorydata7
          out=out_example7 (keep=networkid skuloc echelon period readyrate)
          horizon=8 objective=predictkpi;
run;
```

The original data sets from the section "Two-Echelon Assembly Network" on page 63 are used here along with the newly created inventory data set. The evaluation results are shown in Output 3.25.2.

Output 3.25.2 Output Data Set of the Evaluation with the Independent Solution

An Independent Solution vs. a Joint Solution
Evaluating Silo Solution

Obs	Network ID	Sku Loc	Echelon	Period	Ready Rate
1	N3	FP	1	1	0.89447
2	N3	FP	1	2	0.88945
3	N3	FP	1	3	0.88000
4	N3	FP	1	4	0.88889
5	N3	FP	1	5	0.86432
6	N3	FP	1	6	0.89000
7	N3	FP	1	7	0.87940
8	N3	FP	1	8	0.88500
9	N3	AP	2	1	0.53769
10	N3	AP	2	2	0.55276
11	N3	AP	2	3	0.52764
12	N3	AP	2	4	0.48485
13	N3	AP	2	5	0.51010
14	N3	AP	2	6	0.49495
15	N3	AP	2	7	0.57789
16	N3	AP	2	8	0.53266
17	N3	BP	2	1	0.51759
18	N3	BP	2	2	0.49246
19	N3	BP	2	3	0.50754
20	N3	BP	2	4	0.50505
21	N3	BP	2	5	0.47980
22	N3	BP	2	6	0.47475
23	N3	BP	2	7	0.49749
24	N3	BP	2	8	0.49749

From the output data set, it is quite clear that the independent solution achieves service levels far lower than what is required. The reason is obvious: the policy parameters in the independent solution are much lower than those in the joint solution.

Example 3.26: Intermittent Demand

This example demonstrates how PROC MIRP models the forecast of intermittent demand when the forecast is generated by Croston's method.

Intermittent demand commonly occurs when products are slow-moving, such as spare parts. There are two types of uncertainty in intermittent demand: the time between two demands (*demand interval*) is random, and so is the size of the demand (*demand size*). Croston's method is widely used in forecasting such demand. It assumes that demand occurs as a Bernoulli process, with the demand size from a normal distribution. With these assumptions, it estimates the average demand interval and the mean and variance of the demand size.

To use the forecast generated by Croston's method, you need to specify the average demand interval in the NODEDATA= data set and the mean and variance of demand size in the DEMANDDATA= data set. Consider a single location problem described by the following data sets:

```
data node_croston;
   format networkid $7. skuloc $10.;
   input networkid $8. skuloc $11. leadtime
         servicelevel holdingcost demandinterval;
datalines;
CROSTON SpareParts 1 0.95 1 2
;

data arc_croston;
   format networkid $7. predecessor successor $10.;
   input networkid $8. predecessor $9. successor $11.;
datalines;
CROSTON EXTERNAL SpareParts
;

data demand_croston;
   format networkid $7. skuloc $10.;
   input networkid $8. skuloc $11.
         period mean variance;
datalines;
CROSTON SpareParts 1 1 0
;
```

Notice that the average demand interval is 2, meaning that demand occurs once every two periods on average. When demand occurs, it is always one unit.

The following statements to call PROC MIRP are no different from those used for regular demand patterns. There is no HORIZON= option, so the planning horizon length is set to 12 by default. The POLICY-PARM=INTEGER and DEMANDMODEL=DISCRETE options request that policy parameters be integers and that a discrete distribution be used to model demand. Such settings are recommended because you are dealing with slow-moving items.

```
title 'Intermittent Demand';
proc mirp nodedata=node_croston arcdata=arc_croston
          demanddata=demand_croston out=out_croston
          POLICYPARM=INTEGER
          DEMANDMODEL=DISCRETE
          replications=5000;
run;
```

Output 3.26.1 Output Data Set of the Problem with Intermittent Demand

Intermittent Demand

Obs	Network ID	SkuLoc	Echelon	Period	Reorder Level	Order UpTo Level	Safety Stock	_STATUS_
1	CROSTON	SPAREPARTS	1	1	1	2	1	SUCCESSFUL
2	CROSTON	SPAREPARTS	1	2	1	2	1	SUCCESSFUL
3	CROSTON	SPAREPARTS	1	3	1	2	1	SUCCESSFUL
4	CROSTON	SPAREPARTS	1	4	1	2	1	SUCCESSFUL
5	CROSTON	SPAREPARTS	1	5	1	2	1	SUCCESSFUL
6	CROSTON	SPAREPARTS	1	6	1	2	1	SUCCESSFUL
7	CROSTON	SPAREPARTS	1	7	1	2	1	SUCCESSFUL
8	CROSTON	SPAREPARTS	1	8	1	2	1	SUCCESSFUL
9	CROSTON	SPAREPARTS	1	9	1	2	1	SUCCESSFUL
10	CROSTON	SPAREPARTS	1	10	1	2	1	SUCCESSFUL
11	CROSTON	SPAREPARTS	1	11	1	2	1	SUCCESSFUL
12	CROSTON	SPAREPARTS	1	12	1	2	1	SUCCESSFUL

Output 3.26.1 shows that the reorder and order-up-to levels are 1 and 2, respectively. Stationary forecast of demand is quite common for spare parts. Thus, the policy is stationary. Because the demand during (lead time + 1) periods is one unit on average, the safety stock is one unit ($2 - 1 = 1$). This unit of inventory covers the uncertainty in the demand arrival.

Subject Index

Syntax Index