
# Winklevoss: Pension Functions in R

July 30, 2020

## Chapter 1: Pension Plan Benefits

### I. Retirement Benefit

- Eligibility
  - Normal Retirement: Age 65
  - Early Retirement: Age 55 and 10 years of service
- Benefit
  - 1.5 percent of 5-year average salary per year of service, payable for life, actuarially reduced for early retirement

### II. Vested Benefit

- Eligibility
  - Full vesting after 5 years of service
- Benefit
  - Accrued benefit (based on the retirement benefit formula applied to final average salary and service at termination), payable at age 65 for life

### III. Disability Benefit

- Eligibility
  - Age 40 and 10 years of service
- Benefit
  - Accrued unreduced benefit, payable immediately for life

### IV. Death Benefit

- Eligibility
  - 5 years of service
- Benefit
  - 50 percent of accrued benefit, payable for life of surviving spouse, commencing when employee would have been eligible for early retirement

### V. Employee Contributions

- No vesting/formula

Benefit formula = multiplier * years of service * final average salary

Multiplier = 2%

Years of Service = (60 - 30) = 30

Final Average Salary (5 years of your highest salary): 80k

Annual Benefit = 2% * 30 * $80,000

Annual Benefit = 48,000

What if you have a COLA?

## Chapter 2: Actuarial Assumptions

### Decrement Assumptions

- Decrement Asssumptions:
  - Active members: death, termination, disability, and retirement.
  - Non-active members: death only
  - Single-decrement environment: probability of decrement (q) = rate of decrement (q')
  - Multiple-decrement environment: q (probability) < q' (rate)
    - 2 decrement environment:

$$\mathbf{q^{(1)} = q'^{(1)}[1 - \frac{1}{2}q'^{(2)}]}$$

```
double_decrement <- function(qr_1, qr_2) {

  qr_1 * (1 - (1/2) * qr_2)

}
```

- 3 decrement environment:

$$\mathbf{q^{(1)} = q'^{(1)}[1 - \frac{1}{2}(q'^{(2)} + q'^{(3)}) + \frac{1}{3}q'^{(2)}q'^{(3)}]}$$

```
three_decrement <- function(qr_1, qr_2, qr_3) {

  qr_1 * (1 - ((1/2) * (qr_2 + qr_3)) + ((1/3) * (qr_2 * qr_3)))

}
```

- Mortality Decrement:
  - Probability of a life age $x$ living $n$ years:

$$\mathbf{P_{x \to n} = \prod_{t=0}^{n-1}(1 - q'^{(m)}_{x+t}) = \prod_{t=0}^{n-1}(p'^{(m)}_{x+t})}$$

  - Note that $p$ and $q$ are complementary probabilties of each other

```
# Single decrement

library(readxl)
library(knitr)

dec_tables <- read_excel('Idaho_Decrement_071620.xlsx')

dec_tables <- as.data.frame(dec_tables)

kable(head(dec_tables))
```

| Age | Mortality - Male | Mortality - Female | DR - Male | DR - Female | SR FYE - Male | SR TA - Male | SR FYE - Female | SR TA - Female | ER - Male | ER - Female |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.000637 | 0.000571 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.000430 | 0.000372 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0.000357 | 0.000278 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0.000278 | 0.000208 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0.000255 | 0.000188 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0.000244 | 0.000176 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
survival_prob <- function(start_age = 20, end_age = 65) {
  i <- start_age
  j <- end_age
  prod(1 - dec_tables[i:j, 2])
}

survival_prob(30, 65)
```

```
## [1] 0.8896971
```

```
library(readxl)
library(tidyverse)
```

**You can create multiple decrement column using the previous equations, then simple call that column**

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(knitr)


dec_tables <- read_excel('Idaho_Decrement_071620.xlsx')

dec_tables <- as.data.frame(dec_tables)


kable(head(dec_tables))
```

| Age | Mortality - Male | Mortality - Female | DR - Male | DR - Female | SR FYE - Male | SR TA - Male | SR FYE - Female | SR TA - Female | ER - Male | ER - Female |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.000637 | 0.000571 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.000430 | 0.000372 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0.000357 | 0.000278 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0.000278 | 0.000208 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0.000255 | 0.000188 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0.000244 | 0.000176 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
double_decrement <- function(qr_1, qr_2) {

  qr_1 * (1 - (1/2) * qr_2)

}

three_decrement <- function(qr_1, qr_2, qr_3) {

  qr_1 * (1 - ((1/2) * (qr_2 + qr_3)) + ((1/3) * (qr_2 * qr_3)))

}



male_dec_table <- dec_tables %>%
  select(1:2, 4, 6:7, 10)


male_d_dec <- male_dec_table %>%
  mutate(d_dec = double_decrement(`Mortality - Male`, `DR - Male`))


male_t_dec <- male_d_dec %>%
  mutate(t_dec = three_decrement(`Mortality - Male`, `DR - Male`, `SR TA - Male`))



kable(male_t_dec[40:60,])
```

| | Age | Mortality - Male | DR - Male | SR FYE - Male | SR TA - Male | ER - Male | d_dec | t_dec |
|---|---|---|---|---|---|---|---|---|
| 40 | 40 | 0.001079 | 0.00070 | 0.000 | 0.000 | 0.000 | 0.0010786 | 0.0010786 |
| 41 | 41 | 0.001142 | 0.00078 | 0.000 | 0.000 | 0.000 | 0.0011416 | 0.0011416 |
| 42 | 42 | 0.001215 | 0.00086 | 0.000 | 0.000 | 0.000 | 0.0012145 | 0.0012145 |
| 43 | 43 | 0.001299 | 0.00094 | 0.000 | 0.000 | 0.000 | 0.0012984 | 0.0012984 |
| 44 | 44 | 0.001397 | 0.00102 | 0.000 | 0.000 | 0.000 | 0.0013963 | 0.0013963 |
| 45 | 45 | 0.001508 | 0.00110 | 0.000 | 0.000 | 0.000 | 0.0015072 | 0.0015072 |
| 46 | 46 | 0.001616 | 0.00131 | 0.000 | 0.000 | 0.000 | 0.0016149 | 0.0016149 |
| 47 | 47 | 0.001734 | 0.00152 | 0.000 | 0.000 | 0.000 | 0.0017327 | 0.0017327 |
| 48 | 48 | 0.001860 | 0.00173 | 0.000 | 0.000 | 0.000 | 0.0018584 | 0.0018584 |
| 49 | 49 | 0.001995 | 0.00194 | 0.000 | 0.000 | 0.000 | 0.0019931 | 0.0019931 |
| 50 | 50 | 0.002138 | 0.00215 | 0.000 | 0.000 | 0.000 | 0.0021357 | 0.0021357 |
| 51 | 51 | 0.002449 | 0.00236 | 0.000 | 0.000 | 0.000 | 0.0024461 | 0.0024461 |

| | Age | Mortality - Male | DR - Male | SR FYE - Male | SR TA - Male | ER - Male | d_dec | t_dec |
|---|---|---|---|---|---|---|---|---|
| 52 | 52 | 0.002667 | 0.00257 | 0.000 | 0.000 | 0.000 | 0.0026636 | 0.0026636 |
| 53 | 53 | 0.002916 | 0.00278 | 0.000 | 0.000 | 0.000 | 0.0029119 | 0.0029119 |
| 54 | 54 | 0.003196 | 0.00299 | 0.000 | 0.000 | 0.000 | 0.0031912 | 0.0031912 |
| 55 | 55 | 0.003624 | 0.00320 | 0.220 | 0.100 | 0.030 | 0.0036182 | 0.0034374 |
| 56 | 56 | 0.004200 | 0.00000 | 0.228 | 0.114 | 0.034 | 0.0042000 | 0.0039606 |
| 57 | 57 | 0.004693 | 0.00000 | 0.236 | 0.128 | 0.038 | 0.0046930 | 0.0043926 |
| 58 | 58 | 0.005273 | 0.00000 | 0.244 | 0.142 | 0.042 | 0.0052730 | 0.0048986 |
| 59 | 59 | 0.005945 | 0.00000 | 0.252 | 0.156 | 0.046 | 0.0059450 | 0.0054813 |
| 60 | 60 | 0.006747 | 0.00000 | 0.260 | 0.170 | 0.050 | 0.0067470 | 0.0061735 |

- Salary Assumption = Merit + Productivity + Inflation
- Interest Assumption (Discount Rate, ARR) = Risk-Free Rate + Risk Premium + Inflation

## Chapter 3: Basic Actuarial Functions

**Composite Survival Function:**

$$p_x^{(T)} = (1 - q_x'^{(m)})(1 - q_x'^{(t)})(1 - q_x'^{(d)})(1 - q_x'^{(r)})$$

- Probability of surviving in active service for n years:

$$\mathbf{p}_{\mathbf{n}\to\mathbf{x}}^{(\mathbf{T})} = \prod_{\mathbf{t}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \mathbf{p}_{\mathbf{x}+\mathbf{t}}^{(\mathbf{T})}$$

**Total Number of Employees Leaving Active Service During the Year:**

- $l$ is the notation for the survivors at age x

$$d_x^{(T)} = l_x^{(T)} q_x^{(T)}$$

**Age Functions (reference in downstream equations/functions)**

```
### Chen & Matkin

age_information <<- function(i_ea,retire) {
  ea <<- i_ea # entry age of members,
  # multiple entry ages are not introduced yet

  retire <<- retire  # retirement age

  max_age <<- 85 # maximum age limit

  min_ea<<-30 # minimum entry age

  max_ea<<-45 # maximum entry age

  min_mort_age<<-18 # minimum age in the mortality tables

  max_mort_age<<-120 # maximum age in the mortality tables

  age <<- seq(i_ea,retire)   # age vector from 30-65

  age_xr <<- seq(retire + 1,max_age)  # age vector from 66-100

  yos_xy <<- seq(ea:retire) - 1  # years of service vector

  yos_r_max<<- seq(retire:max_age)-1 # years of service vector
  # from retirenment to max_age

  yos_r_max_mort<<- seq(retire:max_mort_age)-1 # years of service vector from
  # retirenment to max_mort_age

  rx <<- (retire - age)  # future years before retirement vector
```

```
}
```

**Interest Function:**

$$\frac{1}{(1+r)^n}$$

```
# ea <<- i_ea ... entry age of members, multiple entry ages are not introduced yet
# retire <<- retire ... retirement age
# rx <<- (retire - age) ... future years before retirement vector
# i ... interest rate
# max_mort_age<<-120 # maximum age in the mortality tables

## DISCOUNT FACTOR FUNCTIONS

## Discount factor from x to r

get_vrx <- function(ea, retire, i) {
  age_information(ea, retire)
  vrx <- c(1 / ((1 + i) ^ (rx)))
  return(vrx)
}

## Discount factor from x to r & r to 100

get_vxr <- function(ea,retire,i) {

  # generating discount vector for entry age to maximum age of the model(100)

  vxr <-
    c(get_vrx(ea, retire, i),
      c(1 / ((1 + i) ^ (seq((retire + 1):max_mort_age)))))

  return(vxr)

}
```

**Salary Function:**

$$S_x = \sum_{t=y}^{x-1} s_t$$

***Actual Salary***

```
# ea <<- i_ea ... entry age of members, multiple entry ages are not introduced yet
# retire <<- retire ... retirement age
# a_sgr = past salary growth rate
# yos_xy <<- seq(ea:retire) - 1  # years of service vector


get_act_sal <- function(ea, retire, a_sgr) {
```

```
  age_information(ea,retire)

  # actual salary at age x

  return(c((1 + a_sgr) ^ yos_xy))
}
```

***Expected Salary***

```
get_exp_sal <- function(ea,retire,sgr) {
  age_information(ea,retire)
  # expected salary at age x
  return(c((1 + sgr) ^ yos_xy))
}
```

***Accumulated Salary***

```
############### Accumlated Salary ############
get_acc_sal <- function(ea,retire,a_sgr) {
  act_sal <- get_act_sal(ea,retire,a_sgr)
  #acculumated salary at age x
  return(c(cumsum(act_sal)))
}
```

***Expected Salary at Retirement***

```
############### Expected Salary at retirement ############
get_exp_sal_r <- function(ea,retire,a_sgr,sgr) {
  act_sal <- get_act_sal(ea,retire,a_sgr)
  exp_sal_r <- act_sal * ((1 + sgr) ^ rx) # expected salary at age r in year x
  return(exp_sal_r)
}
```

***Salary for Constant Inflation Rate for Different Entry Ages***

```
###############  Salary for constant inflation rate for different entry ages ############
get_sal_vector_ea <- function(ea,retire,sal,inflation,sgr){
  return((((1 + (inflation / 100)) ^ yos_xy)/(1 + ((sgr) / 100)) ^ yos_xy)*sal)
}
```

***Entry Salary from Current Salary for Constant Inflation Rate***

```
############### Entry salary from current Salary for constant inflation rate ############
get_sal_ca_ea<- function(ea,ca,retire,sal,inflation,sgr){
  return(((1 + ((sgr) / 100)) ^ yos_xy[length(age)-(retire-ca)])/((1 + (inflation / 100)) ^ yos_xy[leng
}
```

**Salary at age $x$:**

- $s_y$ = entry-age dollar salary
- $(SS_x)$ = merit salary scale at age x

- $I$ = rate of inflation
- $P$ = rate of productivity reflected in the salary increases.

$$s_x = s_y \frac{(SS_x)}{(SS_y)}[(1+I)(1+P)]^{(x-y)}$$

**Benefit Function**

- Final average salary function:
- $B_r$ = the projected retirement benefit, assuming retirement occurs at the beginning of age
- $r$ = age at retirement
- $n$ = number of years over which the participant's salary prior to retirement is averaged
- $(r-y)$ = years of service
- $k$ = multiplier

$$B_r = k(r-y)\frac{1}{n}\sum_{t=r-n}^{r-1} s_t$$

or more simply

$$B_r = k(r-y)\frac{1}{n}(S_r - S_{r-n})$$

**ACCUMULATED BENEFITS AT RETIREMENT: Chen & Matkin Function**

**There are 16 parameters that can be adjusted in the code.**

1. ENTRY AGE – The starting age of plan participant
2. RETIREMENT AGE – The last year of participant included in payroll
3. CURRENT AGE – The current year of plan participant
4. MINIMUM AGE LIMIT OF THE MODEL – The minimum possible age included in the plan
5. MAXIMUM AGE LIMIT OF THE MODEL – The maximum possible age included in the plan
6. INFLATION RATE – Assumed annual inflation rate
7. PAST SALARYH GROWTH RATE – The actual annual salary growth rate
8. FUTURE SALARY GROWTH RATE – The expected annual salary growth rate
9. DISCOUNT RATE – The assumed annual investment return rate of pension fund
10. BENEFIT FACTOR – The final factor used in pension benefits calculation is a benefit multiplier
11. COLA – The adjustment made to salary to counteract the effects of inflation
12. AFC (AVERAGE FINAL COMPENSATION) – The numbers of last years' salary to be included for calculation
13. VESTING PERIOD – The minimum required years to be qualified for the calculation

14. ACTUARIAL COST METHODS – Actuarial Cost Methods to be entry age normal (EAN) and projected unit calculation (PUC)
15. MORTALITY TABLE CHOICES – Mortality Table choices to be 2(RP 2014), 4(RP 2000), and 6(RP 2010)
16. STARTING SALARY – The starting salary of the first year

```
# ea = entry age
# retire = retirement age
# a_sgr = past salary growth rate
# sgr = future salary growth rate
# afc = average final compensation
# bf = benefit factor


get_acc_benefit_r <- function(ea,retire,a_sgr,sgr,afc,bf) {

# choosing expected salary at retirement vector partitions based on afc period

  if(afc>1)
    exp_sal_r <-get_exp_sal_r(ea,retire,a_sgr,sgr)[1:(length(age) - afc)]

  else
    exp_sal_r <-get_exp_sal_r(ea,retire,a_sgr,sgr)


# storing expected and actual salary vectors

  exp_sal <- get_exp_sal(ea,retire,sgr)

  act_sal <- get_act_sal(ea,retire,a_sgr)


# getting age bounds for averaging actual salary and expected salaries
# in the last afc years

  index_upper <- length(age) - afc

  index_lower <- length(age)


# create an empty vector for accumulated benefits at retirement

  acc_ben_avg <- numeric()
```

```
acc_benefit_r <-

exp_sal_r * (sum((1 + sgr) ^ seq(0,(-afc + 1))) / afc) * bf * (length(age) - 1)


  #creating a accumulated benefit at retirement average for last afc year
  #if afc is greater than 1 year
```

```
  if(afc>1){

    for (i in seq((index_upper+1):index_lower)-1)
      acc_ben_avg <- c(acc_ben_avg,
                       mean(na.omit(c(act_sal[(index_upper+1):(index_upper+1+i)]
                      ,exp_sal[(index_upper+i+2):index_lower])))
                      * bf
                      * (length(age) - 1))

    acc_ben_avg[length(acc_ben_avg)] <-
      mean(act_sal[(index_upper+1):index_lower])
    * bf
    * (length(age) - 1)
  }


  # binding result for accumulated benefits and average for last afc years

  acc_benefit_r <- c(acc_benefit_r,acc_ben_avg)

  return(acc_benefit_r)

}
```

**Annuity Function:**

- **Straight Life Annuity:**

$$\ddot{a}_x = \sum_{t=0}^{\infty} p_{t\to x}^{(m)} v^t$$

```
get_ar <- function(ea,retire,i,cola,mort) {

  # sum of product of discount factor after retirement, probablity
  #of survival after 65, cola and years of service after retirement

  ar <-
    sum(get_vxr(ea, retire, i)
        [(length(get_vrx(ea, retire, i))):length(get_vxr(ea, retire, i))] *

          c(get_rpmx(ea, retire, mort)[length(get_rpmx(ea, retire, mort))],
            na.omit(get_xpmr(ea, retire, mort))) *

          ((1 + cola) ^ yos_r_max_mort))

  return(ar)

}
```

- **Period Certain Life Annuity:**

$$\ddot{a}_{\overline{x:n\urcorner}} = [\sum_{t=0}^{n-1} v^t] + p_{n\to x}^{(m)} v^n \ddot{a}_{x+n}$$

```
get_am <- function(ea, retire, i, amortization) {

  vxr <- get_vxr(ea, retire, i)

  # sum of discount factor from entry age to entry age + amortization period

  am <- sum(vxr[1:(amortization - 1)])

  return(am)

}
```

- **Temporary Annuity**
- Normal Version:

$$\ddot{a}_{\overline{x:n\urcorner}} = \sum_{t=0}^{n-1} p_{t\to x}^{(T)} v^t$$

- Salary-Based Version:

$$\ddot{a}_{\overline{x:n\urcorner}} = \sum_{t=0}^{n-1} \frac{s_t}{s_x} p_{t-x\to x}^{(T)} v^{t-x}$$

One purpose of the temporary annuity is to calculate the actuarial present value of future salary

Normal cost Rate = (PV of Future Benefits) / (PV of Future Salaries)

## Chapter 4: Pension Plan Population Theory

### Basic Concepts

- **Active employees** make up the primary group within the plan's membership, and the following discussion focuses on them.
- **Retired employees** represent another subpopulation, and for present purposes, this group is assumed to consist only of members who retired directly form the service of the employer.
- **Vested terminated** employees make up a third group, which can be further divided into those in the benefit deferral period and those receiving benefits.
- **Disabled employees** make up the fourth sub-population for plans providing disability benefits, and beneficiaries.
- **Surviving spouses** make up a fifth sub-population.

### Stationary Population

A population is considered to be stationary when its size and age distribution remain constant year after year.

### Mature Population

If the increments to the population (newly hired employees) increase at a constant rate, the population will attain a constant percentage age and service distribution in precisely the same length of time as required for a population to become stationary.

### Undermature Population

A population is considered undermature if its age and service distribution has a larger portion of younger, short-service employees than that of a mature population that faces the same decremental factors, is of the same size, and experiences the same entry age distribution.

### Overmature Population

An overmature population is one that has a disproportionately large number of employees at older ages and with longer periods of service than that of a mature population based on the same decrement and entry age assumptions.

### Size-Constrained Population

A size-constrained population will generally converge to its stationary counterpart created without a size constraint. The length of time required for the convergence is a function of the number of ages in the population and the rates of decrements at each of its ages.

Naturally, the more attained ages, other things being equal, the longer it will take for the size-constrained population to come within a predetermined tolerance level of its stationary counterpart.

### Model Plan Populaiton

When dealing with the plan as a whole, it is necessary to assume an age and service distribution of plan members, a specific salary structure of active employees, and a benefit structure of non-active plan members.

In the interest of generality, numerous plan populations, in varying states of maturity, are assumed for the numerical illustrations.

The experience of a single plan population is simulated over a period of 50 years. In order to simulate the various maturity statuses, the initial plan population is undermature and is then forced through a mature state to an overmature status. This is accomplished by first having the size of the population increase at a decreasing rate and eventually decrease at an increasing rate.

### Chen & Matkin R Functions

### Generate Population

```
generate_pop <- function(ea, retire, pop_type, pop_size, median) {

  # calling age information to get age bounds for population
  age_information(ea, retire)

  # generating empty vector of size of age range
  pop1<-numeric(retire - ea + 1)

  # Conditional statements to generate population based on user
  # input for population
  if (pop_type == 'Customize curve')
    pop1 <-round(rtnorm(pop_size, median, 15, left = ea-1, right = retire + 1))

  if (pop_type == 'Over mature')
    pop1 <- round(rtnorm(pop_size, retire, 15, left = ea-1, right = retire + 1))

  if (pop_type == 'Under mature')
    pop1 <- round(rtnorm(pop_size, ea, 15, left = ea-1, right = retire + 1))

  if (pop_type == 'Uniform'){
    repcount <- round(pop_size/(retire-ea))
    for(i in ea:retire)
      pop1<-c(pop1,rep(i,repcount))
    pop1<-pop1[pop1!=0]
  }

  # truncating population exceeding the age range
  pop1<-pop1[pop1!=ea-1]
  pop1<-pop1[pop1!=retire+1]

  # return population vector to server.R
  return(pop1)
}
```

### Generate Retirees Population

```
population_retirees<-function(ea, retire, population){
  # Getting active population frequencies for each age group ###
  pop_freq <- table(population)

  # generating retiree population frequencies for each age after retirement to
  # max_age without considering decline after max age
  pop2 <- rep(pop_freq[length(pop_freq)], max_age - retire)

  # empty vector
  pop3 <- 0

  # age just after retirement ###
  j <- retire + 1

  # translating the retiree frequency to population for age distribution plot
  for(i in pop2){

    pop3 <-c(pop3,rep(j,i))

    j = j + 1

    if(j == max_age + 1)

      break;
  }

  # truncating any active members in the population of retirees
  pop3 <- pop3[2:length(pop3)]

  # return population vector to server.R
  return(pop3)
}
```

### Age Group Generate After Max Age

```
pop_after_max_age <- function(ea, retire){

  age_information(ea, retire)

  return(rep(0, max_mort_age - max_age))
}
```

### Total Population

```
get_total_pop<- function(ea, retire, pop_type, pop_size, median){

  active <- generate_pop(ea, retire, pop_type, pop_size, median)

  retirees <- population_retirees(ea, retire, active)

  after_max_age <- pop_after_max_age(ea, retire)

  return(c(table(c(active, retirees)), after_max_age))
}
```

## Chapter 5: Pension Liability Measures

### General Case

- Actuarial Accrued Liability equals the present value of benefits allocated to date.B' represents the benefits allocated under a given actuarial cost method.

$$ {}^{r}(AL)_x = B'_x {}_{r-x}p_x^{(T)} v^{r-x} \ddot{a}_r $$

### Entry Age Actuarial Liability

- The actuarial accrued liability equals the present value of future benefits, prorated by the salary-based temporary annuity ratio.

$$ {}^{CPr}(AL)_x = \frac{{}^{s}\ddot{a}^{T}_{y:x-y\neg}}{{}^{s}\ddot{a}^{T}_{y:r-y\neg}} B_r {}_{r-x}p_x^{(T)} v^{r-x} \ddot{a}_r $$

$$ {}^{CPr}(AL)_x = \frac{{}^{s}\ddot{a}^{T}_{y:x-y\neg}}{{}^{s}\ddot{a}^{T}_{y:r-y\neg}} {}^{r}(PVFB)_x $$

### Present Value of Future Benefits (from x to r)

```
get_rPVFBx <- function(ea,retire,i,a_sgr,sgr,cola,afc,bf,mort) {
  # product of discount factor to retirement age ,probablity to continue
  # to retirement age,accumulated benefits at 65 and annuity factor for a dollar
  rPVFBx <-
    get_vrx(ea,retire,i) * get_rxpxT(ea,retire,mort) * get_acc_benefit_r(ea,retire,a_sgr,sgr,afc,bf) * g
  return(rPVFBx)
}
```

### Present Value of Term Cost

```
get_PVTC <- function(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting,a){
  age <- age[1:(length(age) - 1)]
  # term cost
  tc <- get_term_cost(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting)
  # creating empty vector for probablity of surviving to next year,
  # discount factor
  np <- numeric(length(age))
  v <- numeric(length(age))
  np <- v <- c(rep.int(0,(length(age))))

  # probablitiy to continue to next year
  xpmx <- get_xpmx(ea,retire,mort)[1:length(age)]

  # computing vectors for probablity of surviving to next year,discount factor
  if (a >= ea) {
    np[a - ea + 1] <- v[a - ea + 1] <- 1
    for (j in (a - ea + 1):(length(age))) {
      np[j + 1] = xpmx[j] * np[j]
```

```
      v[j + 1] = v[j] / (1 + i)
    }
  }

  # PVTC for age 'a'
  pvtc <- sum(np * v * tc)
  return(pvtc)
}

############ PVTC for all ages between entry age and retirement ################
get_PVTC_t <- function(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting){
  age_information(ea,retire)
  pvtc_t <- as.numeric(length(age))
  for (k in 1:(length(age) - 1))
    pvtc_t[k] <-
    c(get_PVTC(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting,age[k]))
  return(c(pvtc_t,0))
}
```

### Term Cost

```
get_term_cost <- function(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting) {
  rpmx <- get_rpmx(ea,retire,mort)
  rpmx <- c(rpmx[2:length(rpmx)],1)
  vc <- get_vesting_cost(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting)
  # product of vesting cost, termination rate,
  # probablity to live to 65 and discount vector
  tc <- vc * get_qxt(ea,retire) * rpmx * get_vrx(ea,retire,i)
  return(tc)
}
```

### Normal Cost (for cost methods)

```
get_NC <- function(ea,retire,i,a_sgr,sgr,cola,afc,bf,cm,mort,vesting) {

  age_information(ea,retire)

  rPVFBx <- get_rPVFBx(ea,retire,i,a_sgr,sgr,cola,afc,bf,mort)

  pvtc <- get_PVTC_t(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting)

  exp_sal <- get_exp_sal(ea,retire,sgr)

  # k is common factor in the NC calculations for both cost methods
  k <-(rPVFBx[1] + pvtc[1]) / (exp_sal[1] * get_tla_t(ea,retire,i,sgr,mort)[1])

  # conditional check and normal cost calculation for different cost methods
  if (cm == 'EAN'){

    # NC formula simulation excel sheet uses PVTC,rPVFBx at entry age
    nc <- (k) * exp_sal
  }
```

```
  else{
    # Normal cost for PUC = (rPVFBx + pvtc) /  years to retirement
    nc <- (rPVFBx+ pvtc) / (age[length(age)] - age[1])
  }

  return(c(nc,rep(0, max_mort_age - retire)))

}
```

### AAL for Cost Methods

```
get_AAL <-
  function(ea, retire, i, a_sgr, sgr, cola, afc, bf, cm, mort, vesting) {

  # load age information
  age_information(ea,retire)
  rPVFBx <- get_rPVFBx(ea,retire,i,a_sgr,sgr,cola,afc,bf,mort)
  pvtc <- get_PVTC_t(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting)

  # k is common factor in the AAL calculations for both cost moethods
  k <- rPVFBx + pvtc

  # conditional check and AAL calculation for different cost methods
  if (cm == 'EAN')

    # AAL for EAN = (rPVFBx + pvtc) - (NC * Temporary annuity)
aal <-
k -

(get_NC(ea,retire,i,a_sgr,sgr,cola,afc,bf,cm,mort,vesting)[1:(retire -ea + 1)] *

c(get_tla_t(ea,retire,i,sgr,mort),
  get_tla(ea,retire,i,sgr,mort,age[length(age)])))

  else

    # AAL for PUC = (rPVFBx + pvtc) * (years of service/ years to retirement)
    aal <- k * ((yos_xy) / (age[length(age)] - age[1]))

  #return(c(aal,get_rPVFBx_after_r(ea,retire,i,a_sgr,sgr,cola,afc,bf,mort)))
return(c(aal,get_rPVFBx_after_r(ea,retire,i,a_sgr,sgr,cola,afc,bf,mort)))

  }
```

## Chapter 6: Normal Costs

- Normal Cost is designed to amortize the PVFB over the employee's working lifetime
- PVFB = PV of future normal costs
- Actuarial Accrued Liability (AAL) = PVFB - PV of future Normal Costs
- Alternatively, AAL = accumulated value of past normal costs
- Entry age normal cost rate K:

The "cost prorate, constant percent" normal cost can be determined by equating the present value of a portion, K, of the participant's future salary tot he present value of future benefits:

$$Ks_y{}^s\ddot{a}^T_{y:r-y\neg} =^r (PVFB)_y$$

$$K = \frac{{}^r(PVFB)_y}{s_y{}^s\ddot{a}^T_{y:r-y\neg}}$$

Then, the normal cost at age $x$ is simply this factor times attained age salary:

$${}^{CPr}(NC)_x = Ks_x$$

- Interesting point: there exists an infinite number of actuarial liabilities and corresponding normal costs

**Present Value for Future Benefits (from current age to retiremetn)**

```
get_rPVFBx <- function(ea, retire, i, a_sgr, sgr, cola, afc, bf, mort) {

  # product of discount factor to retirement age ,probablity to continue
  # to retirement age,accumulated benefits at 65 and annuity factor for a dollar

  rPVFBx <- (

    get_vrx(ea,retire,i) *

    get_rxpxT(ea,retire,mort) *

    get_acc_benefit_r(ea,retire,a_sgr,sgr,afc,bf) *

    get_ar(ea,retire,i,cola,mort)

  )

  return(rPVFBx)
}
```

**Present Value of Term Costs (for a certain age)**

```
get_PVTC <- function(ea, retire, i, a_sgr, cola, afc, bf, mort, vesting, a){

  age <- age[1:(length(age) - 1)]

  # term cost
  tc <- get_term_cost(ea, retire, i, a_sgr, cola, afc, bf, mort, vesting)

  # creating empty vector for probablity of surviving to next year,discount factor
  np <- numeric(length(age))
  v <- numeric(length(age))
  np <- v <- c(rep.int(0, (length(age))))

  # probablitiy to continue to next year
  xpmx <- get_xpmx(ea, retire, mort)[1:length(age)]

  # computing vectors for probablity of surviving to next year,discount factor
  if (a >= ea) {
    np[a - ea + 1] <- v[a - ea + 1] <- 1
    for (j in (a - ea + 1):(length(age))) {
      np[j + 1] = xpmx[j] * np[j]
      v[j + 1] = v[j] / (1 + i)
    }
  }

  # PVTC for age 'a'
  pvtc <- sum(np * v * tc)
  return(pvtc)
}
```

**Expected Salary (at age x)**

```
get_exp_sal <- function(ea,retire,sgr) {

  age_information(ea,retire)

  # expected salary at age x
  return(c((1 + sgr) ^ yos_xy))
}
```

**Normal Cost (cost methods)**

```
get_NC <- function(

  ea, retire, i, a_sgr, sgr, cola, afc, bf, cm, mort, vesting

  ) {

  age_information(ea, retire)

  rPVFBx <- get_rPVFBx(ea, retire, i, a_sgr, sgr, cola, afc, bf, mort)

  pvtc <- get_PVTC_t(ea, retire, i, a_sgr, cola, afc, bf, mort, vesting)
```

```
  exp_sal <- get_exp_sal(ea, retire, sgr)

  # k is common factor in the NC calculations for both cost moethods
  k <- (rPVFBx[1] + pvtc[1]) /
    (exp_sal[1] * get_tla_t(ea, retire, i, sgr, mort)[1])

  # conditional check and normal cost calculation for different cost methods
  if (cm == 'EAN'){

    # NC formula simulation excel sheet uses PVTC,rPVFBx at entry age
    nc <- (k) * exp_sal
  }

  else{
    # Normal cost for PUC = (rPVFBx + pvtc) /  years to retirement
    nc <- (rPVFBx + pvtc) / (age[length(age)] - age[1])
  }

  return(c(nc, rep(0, max_mort_age - retire)))
}
```

## Chapter 8: Ancillary Benefits

1. Vested Termination Benefits
2. Disability Benefits
3. Surviving Spouse Benefits
4. Ancillary Benefits Under Actuarial Cost Methods
5. Cost Prorate Methods
6. Ancillary Benefits for Alternative Liability Measures
7. Comparison of Ancillary Benefit Costs

### Term Cost Concept

- If the benefit under consideration is a lump sum death benefit, the term cost is equal to the cost of one year's term insurance in the amount of the death benefit.
- Term cost is used in this chapter as a precursor to determining the cost of ancillary benefits under various cost methods.

### Vested Termination Benefits

#### Term Cost

The term cost (TC) of vested termination benefits for an employee age $x$ is given by:

- $g_x^{(v)}$ = grading function equal to the proportion of accrued benefit vested at age $x$
- $B_x$ = accrued benefit at age $x$ as defined by the plan benefit formula
- $q_x^{(t)}$ = probability of terminating during age $x$
- ${}_{r-x-1}p_{x+1}^{(m)}$ = probability of living from age $x$ + *1* to retirement
- $v^{r-x}$ = discount factor to age x
- $\ddot{a}_r$ = straight life annuity

$$ {}^{v}(TC)_x = g_x^{(v)} B_x q_x^{(t)} {}_{r-x-1}p_{x+1}^{(m)} v^{r-x} \ddot{a}_r $$

This formulation shows that the term cost of vesting is the expected liability associated with the contingency that the employee may terminate vested during age x . . . *the grading function and accrued benefit function could be evaluated at age x + 1/2 to approximate a fractional year's credit. Similarly, the survival function could be evaluated at this fractional age as well.*

***Grading Function***

```
# ea = entry age
# retire = retirement age
# vesting = ??vesting period in years??


get_gxv <- function(ea, retire, vesting) {

  ### repeat zeros for vesting period and a 100 from vesting + 1 to retirement
```

```
  gxv <- c(rep(0,vesting), rep(100,(length(age) - vesting)))

  return(gxv)

  }
```

**Straight Life Annuity**

```
get_ar <- function(ea,retire,i,cola,mort) {

  # sum of product of discount factor after retirement, probability
  #of survival after 65, cola and years of service after retirement

  ar <-
    sum(get_vxr(ea, retire,i)[(length(get_vrx(ea, retire,i))):
                                length(get_vxr(ea,retire,i))] *
          c(get_rpmx(ea, retire, mort)[length(get_rpmx(ea,retire, mort))],
            na.omit(get_xpmr(ea,retire,mort))) *
          ((1 + cola) ^ yos_r_max_mort))

  return(ar)
}
```

***Discount Factors***

```
## Discount factor from x to r

get_vrx <- function(ea, retire, i) {
  age_information(ea, retire)
  vrx <- c(1 / ((1 + i) ^ (rx)))
  return(vrx)
}

## Discount factor from x to r & r to 100

get_vxr <- function(ea, retire, i) {

  # generating discount vector for entry age to maximum age of the model(100)

  vxr <-
    c(get_vrx(ea, retire, i),
      c(1 / ((1 + i) ^ (seq((retire + 1):max_mort_age)))))

  return(vxr)

}
```

***Vesting Cost***

```
# ea = entry age
# retire = retirement age
# i = discount rate
# a_sgr = actual salary growth rate
```

```
# cola = COLA
# afc = average final compensation
# bf = benefit factor
# mort = ????
# vesting = ??vesting period in years??
# get_ar = straight life annuity factor for 1 dollar

get_vesting_cost<- function(ea,retire, i, a_sgr, cola, afc, bf, mort, vesting) {

  bx <- get_acc_benefits(ea,retire, a_sgr, afc, bf)

  # product of grading function, accumulated benefits, and annuity factor

  vc <- (get_gxv(ea,retire,vesting) / 100) * bx * get_ar(ea, retire,i,cola,mort)

  return(vc)

}
```

***Termination Rates***

```
get_qxt <- function(ea,retire) {

  # load age information
  age_information(ea,retire)


  # minumum age in the 2003 SOA Pension Plan Turnover Study (the select and ultimate table)

  min_qxt_age<<-18


  # empty vector of numeric type

  term_rate<-numeric()


  # creating termination rate vector based on entry age and years of service

  for(i in yos_xy){
    if(yos_xy[i+1]<2)
      term_rate<-c(term_rate,term_rate_xl$`Service<2`[ea-min_qxt_age+1+i]/100)
    else if(yos_xy[i+1]>=2 && yos_xy[i+1]<=4)
      term_rate<-c(term_rate,term_rate_xl$`Service=2,3,4`[ea-min_qxt_age+1+i]/100)
    else if(yos_xy[i+1]>=5 && yos_xy[i+1]<=9)
      term_rate<-c(term_rate,term_rate_xl$`Service=5-9`[ea-min_qxt_age+1+i]/100)
    else
      term_rate<-c(term_rate,term_rate_xl$`Service>=10`[ea-min_qxt_age+1+i]/100)
  }

  term_rate[is.na(term_rate)]<-0

  return(term_rate)
```

```
}
```

***Survival Probability from x to Retirement Age***

```
get_rpmx <- function(ea, retire, mort) {

  xpmx <- get_xpmx(ea, retire, mort)

  ### generating vector by reversing original xpmx vector and taking a
  ### cumulative product and reversing this result again

  rpmx <- c(rev(cumprod(rev(xpmx[1:(length(age) - 1)]))), 1)

  return(rpmx)

}
```

***Term Cost***

```
get_term_cost <- function(ea, retire, i, a_sgr, cola, afc, bf, mort, vesting) {

  rpmx <- get_rpmx(ea, retire, mort)

  rpmx <- c(rpmx[2:length(rpmx)], 1)

  vc <- get_vesting_cost(ea, retire, i, a_sgr, cola, afc, bf, mort, vesting)

  # product of vesting cost, termination rate,
  # probability to live to 65, and discount vector

  tc <- vc * get_qxt(ea,retire) * rpmx * get_vrx(ea,retire,i)

  return(tc)

}
```

**Present Value of Future Vested Termination Benefits**

The PVFB for vested termination benefits can be expressed by taking the present value of the employee's future term cost of vesting:

$$ {}^{v}(PVFB)_x = \sum_{k=x}^{r'-1} {}_{k-x}p_x^{(T)} v^{k-x\,v}(TC)_k $$

***Present Value Term Cost for a Certain Age***

```
get_PVTC <- function(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting,a){
  age <- age[1:(length(age) - 1)]
  # term cost
  tc <- get_term_cost(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting)
  # creating empty vector for probablity of surviving to next year,discount factor
  np <- numeric(length(age))
```

```
  v <- numeric(length(age))
  np <- v <- c(rep.int(0,(length(age))))

  # probablitiy to continue to next year
  xpmx <- get_xpmx(ea,retire,mort)[1:length(age)]

  # computing vectors for probablity of surviving to next year,discount factor
  if (a >= ea) {
    np[a - ea + 1] <- v[a - ea + 1] <- 1
    for (j in (a - ea + 1):(length(age))) {
      np[j + 1] = xpmx[j] * np[j]
      v[j + 1] = v[j] / (1 + i)
    }
  }

  # PVTC for age 'a'
  pvtc <- sum(np * v * tc)
  return(pvtc)
}
```

***PVTC for All Ages Between Entry Age and Retirement***

```
get_PVTC_t <- function(ea,retire,i,a_sgr,cola,afc,bf,mort,vesting){
  age_information(ea,retire)
  pvtc_t <- as.numeric(length(age))
  for (k in 1:(length(age) - 1))
    pvtc_t[k] <-
    c(get_PVTC(ea, retire, i, a_sgr, cola, afc, bf,mort, vesting, age[k]))
  return(c(pvtc_t,0))
}
```

**Disability Benefits**

**Term Cost**

The term cost (TC) of disability benefits for an employee age $x$ is given by

- $g_x^{(v)}$ = grading function equal to the proportion of accrued benefit vested at age $x$
- $B_x$ = accrued benefit at age $x$ as defined by the plan benefit formula
- $q_x^{(t)}$ = probability of disabled during age $x$
- $w$ = waiting period before disability benefits commence
- ${}_w^d p_{x+1}^{(m)}$ = probability that a disabled life age x lives w years
- $\ddot{a}_{x+w}^d$ = life annuity based on disabled-life mortality
- $v^{w+1}$ = discount factor to age w

$$^d(TC)_x = g_x^{(d)} B_x q_x^{(d)} {}_w^d p_{x+1}^{(m)} v^{w+1} \ddot{a}_{x+w}^d$$

The term cost of disability will be zero prior to the first qualification age and will generally increase thereafter, although this need not be the case under some benefit formulas or actuarial assumptions.

**Present Value of Future Disability Benefits**

$$^d(PVFB)_x = \sum_{k=x}^{r'-1} {}_{k-x}p_x^{(T)} v^{k-x} {}^d(TC)_k$$

**Surviving Spouse Benefits**

Similar equations

### Ancillary Benefits Under Actuarial Cost Methods

The total cost of both ancillary and retirement benefits under various individual actuarial cost methods is considered in this section. The aggregate versions of each cost method are not presented, but the principles are the same.

**Accrued Benefit Method** The normal cost under the accrued benefit method, consistent with the underlying theory of this method, is equal to the present value of a deferred annuity of $b_x$ (the formula benefit accrual), where the present value is based on the possibility that the employee may either terminate vested and be entitle to a deferred vested benefit, become disabled and receive a disability benefit, die and leave a spouse with a benefit, or retire and recieve a retirement benefit.

The total normal cost for a participant age x can be represented by

_{k-x}p_x^{T} v^{k-x} ({q}_k^{(t)} ^vF_k)

### Benefit Prorate Methods

### Cost Prorate Methods

### Ancillary Benefits for Alternative Liability Measures

### Comparison of Ancillary Benefit Costs