

생존분석 보고서

유재성¹

최종 수정일자: 2018년 11월 18일

프로토타입 링크: https://github.com/praster1/Note_SurvivalAnalysis

Contents

I 서론	5
1 서론	5
2 임상적 연구 결과 데이터	5
II 생존 자료의 분석	6
3 생존 분석 방법의 분류	8
III 생존 함수	9
4 반응변수	9
5 생존 함수(survival function), 신뢰도 함수(reliability function)	10
6 위험 함수(hazard function), 위험률 함수(hazard rate function; 고장률 함수 failure rate function)	11
7 누적 위험 함수(cumulative hazard function), 누적 고장률 함수(cumulative failure rate function)	12
8 생존 함수간 관계	12
9 위험률 함수와 생존함수의 1:1 대응	13
9.1 역공식을 이용한 위험함수와 생존함수의 관계	13
9.2 분포의 비모수적 분류	15
9.2.1 비모수적 군	15
9.2.2 비모수적 군들의 상호관계	16
IV 척도 함수	17
10 '고장(재발) 발생까지 시간 길이의 길고 짧음'에 따른 척도 함수	17
10.1 생존 함수(Survival Function)	17
10.2 평균 수명(mean time to failure)	17
10.3 평균 고장 간격(mean time between failure)	18
10.4 백분위수	18
10.5 Bearing Life	18
11 '고장(재발) 발생까지 시간 길이의 길고 짧음'에 따른 척도 함수	19
11.1 평균 잔여 수명 함수	19
11.2 가용도 함수	20
11.3 위험률(고장률)	21
11.3.1 비수리계(non-repairable system)의 위험률(고장률)	21
11.3.2 수리계(repairable system)의 위험률(고장률)	21
11.3.3 평균 위험률(평균 고장률, average failure rate)	21
V 수명 자료(Life Data), 중도 절단 자료(Censored Data)	22
12 우중도절단(right censoring)	24
12.1 완전 자료(complete data)	25
12.2 Type 1 우중도절단 자료(정시종단자료)	25
12.3 Type 2 우중도절단 자료(정수종단자료)	25
12.4 임의 우중도절단 자료	26
12.4.1 Kaplan-Meier Estimation, Product Limit Estimation	26
12.4.2 Life-table Method	27
13 Truncation	28

¹고려대학교 컴퓨터학과 박사과정. lv999@korea.ac.kr

VI 모수적 방법을 이용한 생존함수의 추정

29

14 확률분포	29
14.1 Alpha Distribution(알파 분포)	31
14.2 Arcsine Distribution	35
14.3 Beta Distribution(베타 분포)	39
14.4 Birnbaum-Saunders Distribution	43
14.5 Burr Distribution of Type XII	47
14.6 Cauchy Distribution	51
14.6.1 Half-Cauchy Distribution	55
14.7 Chi(χ) Distribution	58
14.8 Chi-square(χ^2) Distribution(카이제곱 분포)	61
14.9 Cosine Distribution	64
14.9.1 Ordinary Cosine Distribution	64
14.9.2 Raised Cosine Distribution	68
14.10 Dhillon's Distribution(딜론 분포)	72
14.10.1 Dhillon's I Distribution	73
14.10.2 Dhillon's II Distribution	77
14.11 Exponential Distribution(지수 분포)	81
14.11.1 Exponential Distribution with Location Parameter	89
14.11.2 Exponentiated Exponential Distribution	93
14.11.3 Reflected Exponential Distribution	96
14.12 Extreme Value Distribution(극치 분포)	100
14.12.1 최대극치분포(The Maximum Extreme Value Distribution)와 최소극치분포(The Minimum Extreme Value Distribution)	100
14.12.2 Type I 최소값 극치분포(Gumbel 최소값 분포)	101
14.12.3 Type I 최대값 극치분포(Gumbel 최대값 분포)	105
14.12.4 Type II 극치분포(Frechet 분포)	109
14.12.5 Type II 극치분포(Frechet 분포) with Location Parameter	112
14.13 F Distribution	115
14.14 Gamma Distribution(감마 분포)	119
14.14.1 Gamma Distribution with 2 Parameters(2-모수 감마 분포)	119
14.14.2 Gamma Distribution with Location Parameter	128
14.14.3 Log-gamma Distribution	131
14.14.4 Generalized Gamma Distribution	132
14.15 Generalized Exponential Geometric Distribution	133
14.16 Gompertz Distribution	134
14.17 Gompertz-Makeham Distribution	138
14.18 Hjorth Distribution(호스 분포)	142
14.19 Hyperbolic Secant Distribution	146
14.20 Laplace Distribution	150
14.20.1 Log-Laplace Distribution	154
14.21 Linear Hazard Rate Distribution(선형 증가 분포)	155
14.21.1 Generalized Linear Hazard Rate Distribution	159
14.22 Logistic Distribution	160
14.22.1 Log-logistic Distribution	164
14.22.2 Half-logistic Distribution	168
14.22.3 Generalized Logistic Distribution	172
14.23 Lomax Distribution	173
14.23.1 Generalized Lomax Distribution	177
14.24 Makeham Distribution(메이크햄 분포)	181
14.25 Maxwell-Boltzmann Distribution	184
14.26 Muth Distribution	185
14.27 Normal(Gaussian) Distribution	189
14.27.1 Log-normal Distribution(대수 정규 분포, 로그 정규 분포)	193
14.27.2 Log-normal Distribution with Lower Threshold	197
14.27.3 Log-normal Distribution with Upper Threshold	198
14.27.4 Inverse Normal(Gaussian) Distribution	199
14.27.5 Half-normal Distribution	202
14.27.6 Trimmed(Truncated) Normal Distribution(절사 정규 분포)	205
14.28 Parabolic U-shaped Distribution	206
14.28.1 Parabolic Inverted U-shaped Distribution	210
14.29 Pareto Distribution of the first kind	214
14.29.1 Generalized Pareto Distribution of the first kind	217
14.30 Power Function Distribution	218
14.31 Rayleigh Distribution(레이리 분포)	222
14.31.1 Inverse Rayleigh Distribution	225
14.31.2 Generalized Rayleigh Distribution	229
14.32 Semi-elliptical Distribution	230
14.33 t Distribution	234
14.34 Teisser Distribution	237
14.35 Triangular Distribution(Continuous)	241
14.36 Uniform Distribution(Continuous)	242
14.37 V-shaped Distribution	246
14.38 Wald Distribution	246
14.39 Weibull Distribution(와이블 분포)	247
14.39.1 Weibull Distribution with 2 Parameters(2-모수 와이블 분포)	247
14.39.2 Weibull Distribution with 3 Parameters(3-모수 와이블 분포)	254
14.39.3 Log-Weibull Distribution	258
14.39.4 Double Weibull Distribution	262
14.39.5 Inverse Weibull Distribution	266
14.39.6 Reflected Weibull Distribution	270
14.40 Wigner's Semi-circle Distribution	273

VII Cox 비례위험모형

274

15 Covariate가 1개인 비례위험모형	275
15.1 데이터 구조 - covariate가 1개인 경우	275
15.2 준모수적 Cox 비례위험모형	275
15.3 회귀계수에 대한 추론	276
16 Covariate가 여러 개인 비례위험모형	278
16.1 데이터 구조 - covariate가 여러 개인 경우	278
16.2 준모수적 Cox 비례위험모형	278
16.3 회귀계수와 누적위험함수의 추정	278
16.3.1 비례위험모형	279
16.3.2 부분가능도함수	279
16.3.3 정보행렬(information matrix)	280

17 동점 처리(handling ties)	281
17.1 Exact Method	281
17.2 Breslow's Approximation	282
17.3 Efron's Approximation	283
VIII Performance Evaluation Metrics	284
18 반응 변수가 binary data인 경우	285
18.1 Receive operating characteristic curve(ROC Curve) & Area under the curve(AUC)	285
18.2 Net reclassification improvement(NRI)	286
18.3 Mean Square Error(MSE) & Mean Absolute Error(MAE)	287
18.4 Cox and Snell의 결정계수(R^2) & Nagelkerke의 결정계수(R^2)	287
18.5 Brier Score	287
18.6 Hosmer-Lemeshow Test	287
19 반응 변수가 생존 시간인 경우	288
19.1 Ignoring time to event (logistic C)	288
19.2 Harrell의 Concordance Index(C-index)	288
19.3 Heagerty의 integrated AUC	289

IX Results of Survey	290
20 Neural Network with Life Data	291
20.1 A practical application of neural network analysis for predicting outcome	291
20.1.1 의견	291
20.2 Survival analysis and neural nets	292
20.2.1 의견	293
20.3 A Neural Network Model for Survival Data	294
20.3.1 의견	294
20.4 On the use of artificial neural networks for the analysis of survival data	295
20.4.1 의견	297
20.5 Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach	298
20.6 Comparison of the performance of neural network methods and Cox regression for censored survival data	301
20.6.1 데이터셋	301
20.6.2 요약	302
20.6.3 의견	302

List of Codes

Code 1: Alpha Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	32
Code 2: Arcsine에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	36
Code 3: Beta Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	40
Code 4: Birnbaum-Saunders Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	44
Code 5: Burr Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	48
Code 6: Cauchy Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	52
Code 7: Half-Cauchy Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	56
Code 8: Chi Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	59
Code 9: Chi-square Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	62
Code 10: Cosine Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	65
Code 11: Raised Cosine Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	69
Code 12: Dhillon's I Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	74
Code 13: Dhillon's II Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	78
Code 14: Exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	82
Code 15: Exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	90
Code 16: Exponentiated exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	93
Code 17: Reflected Exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	97
Code 18: Type I 극치분포(Gumbel 최소값 분포) Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	102
Code 19: Type I 극치분포(Gumbel 최대값 분포) Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	106
Code 20: Type II 극치분포(Frechet 최대값 분포)에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	109
Code 21: Type II 극치분포(Frechet 분포) with Location Parameter에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	112
Code 22: F Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	116
Code 23: 2모수 감마 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	120
Code 24: 3모수 감마분포에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	128
Code 25: Gompertz Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	135
Code 26: Gompertz-Makeham Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	139
Code 27: Hjorth Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	143
Code 28: Hyperbolic Secant Distribution Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	147
Code 29: Laplace Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	151
Code 30: 선형증가 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	156
Code 31: Logistic Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	161
Code 32: Log-logistic Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	165
Code 33: Half-logistic Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	169
Code 34: Lomax Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	174
Code 35: Generalized Lomax Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	178
Code 36: Makeham Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	182
Code 37: Muth Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	186
Code 38: Normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	190
Code 39: Log-normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	194
Code 40: Inverse Normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	200
Code 41: Half-normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	203
Code 42: 절사정규분포에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	205
Code 43: Parabolic U-shaped Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	207
Code 44: Parabolic Inverted U-shaped Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	211
Code 45: Pareto Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	215
Code 46: Power Function Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	219

Code 47: Rayleigh Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	223
Code 48: Inverse Rayleigh Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	226
Code 49: Semi-elliptical Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	231
Code 50: t Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	235
Code 51: Teisser Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	238
Code 52: Uniform Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	243
Code 53: 2모수 와이블 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	248
Code 54: 3모수 와이블 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	255
Code 55: Log-Weibull Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	259
Code 56: Double Weibull Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	263
Code 57: Inverse Weibull Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	267
Code 58: Alpha Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)	271

Part I

서론

1 서론

생존분석(survival analysis)에서는, 어떤 연구에 들어온 시간부터 어떤 사건이 발생할 때까지의 시간 구간(time interval) 데이터에 관심이 있다.

관심 있는 **반응변수**는, 사건이 발생할 때까지 걸리는 시간이다.

생존분석의 주 관심사는 다음 두 가지로 요약할 수 있다.

- 생존함수(survival function)의 추정
 - 생존함수(survival function) 또는 위험함수(hazard function)에 영향을 주는 공변량(covariate) 또는 예측변수를 찾아내어 그 연관 정도를 추정
- ***공변량(covariate)** 또는 **예측변수** :
- 일반적으로 연구자가 통제할 수 있는 처리(treatment)
 - 또는 실제 관심요인은 아니지만, 반응변수에 유의한 영향을 줄 수 있는 요인
-

생존분석이 다른 통계분석방법들과 구별되는 가장 큰 특징은, 중도절단자료(censored data)를 포함한다는 것이다.
생존분석 자료는 유한한 시간 내에서 일어나는 사건만 측정할 수 있다.

2 임상적 연구 결과 데이터

임상 연구는 크게 laboratory research와 clinical research로 분류되고, 송경일과 최종수(2007)[2]에 따르면, clinical research는 다시 다음과 같이 분류된다.

- **Experimental Study**에서는 대상자에서 약물 투여와 같은 treatment 혹은 intervention 후 그 효과에 대해 연구한다. 흔히 longitudinal study이며, intervention한 후 일정한 시간이 지난 후 반응을 연구한다.
- **Observation Study**는 연구자가 대상자의 존재하는, 혹은 기존에 존재하고 있는 상황을 관찰하고 기술하고 분석한다.
 - **Cohort Study**는 특정 연구대상을 선정하고 그 대상으로부터 특정 질병의 발생에 관여하리라 의심되는 어떤 특성이나 위험인자에 폭로된 정보를 수집한 후, 특정 질병의 발생을 시간 경과에 따라 prospective(전향적으로)하게 추적조사 혹은 관찰함으로써 특정 요인에 폭로되지 않은 집단에 비해 폭로된 집단에서의 질병의 발생률을 비교하는 방법이다.
 - **Case Control Study(환자대조군 연구)**는 후향적 연구이며, 관찰된 결과를 관심있는 사건이 발생한 case와 발생하지 않은 control로 나누어서, 사건의 결과를 발생하게 한 가능한 위험인자를 찾아내는 방법이다.
 - **Cross Sectional Study(단면조사 연구)**는 현재 연구시점의 한 시점에서 일정한 인구 집단을 대상으로 모집단을 대표할 수 있는 표본을 추출하여 조사한 후, 속성에 따라 이들 표본집단을 의심되는 원인에 폭로된 집단과 의심되는 원인에 폭로되지 않은 집단으로 나누어 차이점을 분석하는 것이다.

Part II

생존 자료의 분석

생존 분석에 사용되는 생존 자료는 다음과 같은 특징이 있다.

- 생존과 관계되는 결과변수는 '사망' 혹은 생존'과 같은 생존의 여부(binary outcome)와, 생존 시간(survival time)이라는 두 가지의 형태로 이루어져 있다.
- 생존 시간은 음수(negative)로 측정될 수 없고, 향상 양수(positive) 값으로 측정된다.
- 사건이 발생할 때까지 걸리는 시간인 'time to event occurrence'에 대한 분포는 보통 정규분포를 하지 않는다. 정규분포를 하지 않는 이유는, 결과가 발생하기까지의 시간 간격이 대상자마다 다르고, 연구가 종료될 때 모든 대상자에서 사건이 발생하지 않기(다양한 이유의 censoring) 때문이다.
- 중도 절단된 관찰(censored observation)이 발생하기 마련이다.

위와 같은 생존 자료의 특성상, 생존 자료는 일반적인 통계 분석 방법으로는 분석하기 어렵다. 그럼 어떤 방법으로 분석하는 것이 좋을까?

- multiple linear regression으로 분석한다면 어떨까?
 - 그러나, 생존 자료의 종속 변수는 대부분 정규분포를 하지 않기 때문에² multiple linear regression에서 독립변수의 분포는 정규분포를 한다는 가정에 위배된다.
 - 또한, 중도 절단이 발생하므로, multiple linear regression은 중도 절단에 대한 처리 방법이 없으며, 이분형의 결과 변수를 처리할 수 없으므로 생존 자료는 분석하기 어렵다.
 - 만약 중도 절단된 자료가 전혀 없고, 모든 대상자에서 사건이 발생했다면, 그리고 생존시간도 어느 정도 정규분포를 한다면, 예측 인자와 time to event의 사이의 분석에 multiple linear regression을 이용할 수도 있다.
- binary logistic regression으로 분석한다면 어떨까?
 - logistic regression analysis도 중도 절단된 자료를 처리할 수 없으며, time to event에 대한 자료를 분석할 수 없다.
 - logistic regression은 '사건의 발생 여부'에 초점을 둔 방식이고, 중도 절단이 없다면 logistic regression을 할 수 있다. 하지만 중도 절단된 자료도 분명히 생존시간에 기여를 했는데, 이를 분석에서 제외한다면, 중요한 정보를 소실하게 되는 것이다.

방법	Linear Regression	Logistic Regression	Survival Analysis
예측 변수	Cartegorical or Continuous	Cartegorical or Continuous	Time and {Cartegorical or Continuous}
결과 변수	Normal Distribution	Binary	Binary
중도 절단 허용	No	No	Yes
수학적 모형	$Y = B_1 X + B_0$	$\log\left(\frac{p}{1-p}\right) = B_1 X + B_0$	$H(t) = H_0(t) \exp^{B_1 X + B_0}$
결과의 산출	Linear Change	Odds Ratios	Hazards Rates

Table 1: Regression과 생존분석의 차이점

생존 분석을 하는 목적은, 주어진 생존 자료로 다음과 같은 결과를 보고자 함이다.

- descriptive survival analysis: 생존함수³를 추정하여 모집단에서 생존 시간의 분포를 알아보기 위해서 (예: 1 year survival rate, median survival time 등)
- univariate survival analysis: 추정된 생존함수가 집단에 따라서 차이가 있는지를 비교하기 위해서 (예: 치료집단과 비치료집단 사이에 생존율의 차이가 있는가?)
- semi-parametric survival analysis⁴: Time to event와 이에 영향을 미치는 독립변수들과의 관계를 찾기 위해서 (예: 생존율이 차이가 있다면, 생존율에 영향을 미치는 중요한 인자는 무엇인가?)

²생존시간은 보통 exponential, Weibull, log normal distribution을 따른다

³"생존 분석"에서는 $S(t)$ "신뢰성 공학"에서는 $R(t)$ 로 주로 표현한다.

⁴cox-proportional hazards model 포함

생존 분석의 기본 가정은 다음과 같다. 데이터가 다음의 기본적인 가정을 만족하지 못하면, 생존분석을 수행할 수 없다.

- 생존분석에서 대상자에 대해 관찰을 시작하고 종료하는 것은, 생존 여부의 발생에는 영향을 주지 않는다.
즉, 생존 여부의 발생은 완전히 무작위로 발생해야 하며, 생존의 간으성은 연구기간 동안 내내 항상 일정하다.
- 추적조사에서 탈락된 대상자나 연구에 끝까지 참여한 대상자나 같은 예후를 가진다.
즉, 생존분석에서 중도 절단의 발생은, 시간 경과에 따른 생존 여부와는 완전히 무관하게 무작위로 발생하여야(random censoring) 하며, 생존시간과는 독립적이어야 한다.
- 각 연구 대상자가 중도 절단 될 확률은, 사건이 발생한 연구 대상자가 경험할 확률과는 관계가 없다.
- 비례위험 모형에 한하여, hazards에 영향을 미치는 어떤 요인은, 연구기간 동안 항상 같은 비율로 작용한다.

Note 1. 생존 분석의 주요 역사[48]

- 1950~1960년대 임상 데이터가 폭발적으로 증가 & 발전하던 시기임.
 - 1969, 1972 Cumulative Hazard Function의 non-parametric 추정량이 Nelson에 의해서 처음으로 제안됨.
 - 1970년대 노르웨이의 통계학자들과 버클리의 통계학자들이 공동연구를 많이 함.
 - 1972 1972년에 Aalen은 석사학위논문을 통해서 Nelson과 똑같은 추정량을 석사학위논문을 통해 제안.
 - 1973 Kaplan Meier 추정량의 matrix 버전이 소개됨.
 - 1974 Breslow에 의해 Cumulative baseline hazard rate가 추정되었다.
 - 1975 Aalen은 박사학위논문을 통해 마팅게일과 더불어 hazard function과 현재의 Nelson-Aalen 추정량이라 불리우는 형태를 소개함. (Nelson이 1969년의 그 Nelson이다)
 - 1976 Aalen이 마팅게일로써의 two-sample test를 소개함.
 - 1978 Aalen은 위에서 weight function이 적용된 특별한 경우의 test score를 보여줌.
코펜하겐에서 recurrent event analysis에 관한 프로젝트가 시작되었다.
 - 1979~1980 Fleming에 의해, 마팅게일 카운팅 어프로치를 바탕으로 한 Kaplan-Meier 추정량이 발표됨.
 - 1980년대 Aalen이 additive hazards model을 제공하기 시작했다.
 - 1981 Per Kragh Anderson에 의해 Cox score function과 다양한 접근 속성을 위한 마팅게일 성질이 적용되기 시작했다.
(헷갈릴 수 있는데, Anderson-Darling의 Theodore Wilbur Anderson과 다른 사람이다.)
 - 1981 Time invariant covariates의 경우 large sample properties를 다루는 전통적인 모델이 제공되기 시작함.
 - 1985 Anderson과 Bogan에 의해 multivariate counting processes modeling으로 Cox model이 확장됨.
 - 1988, 1990 Goodness-of-fit test를 기반으로 한 Martingale residual이 정의됨
-

3 생존 분석 방법의 분류

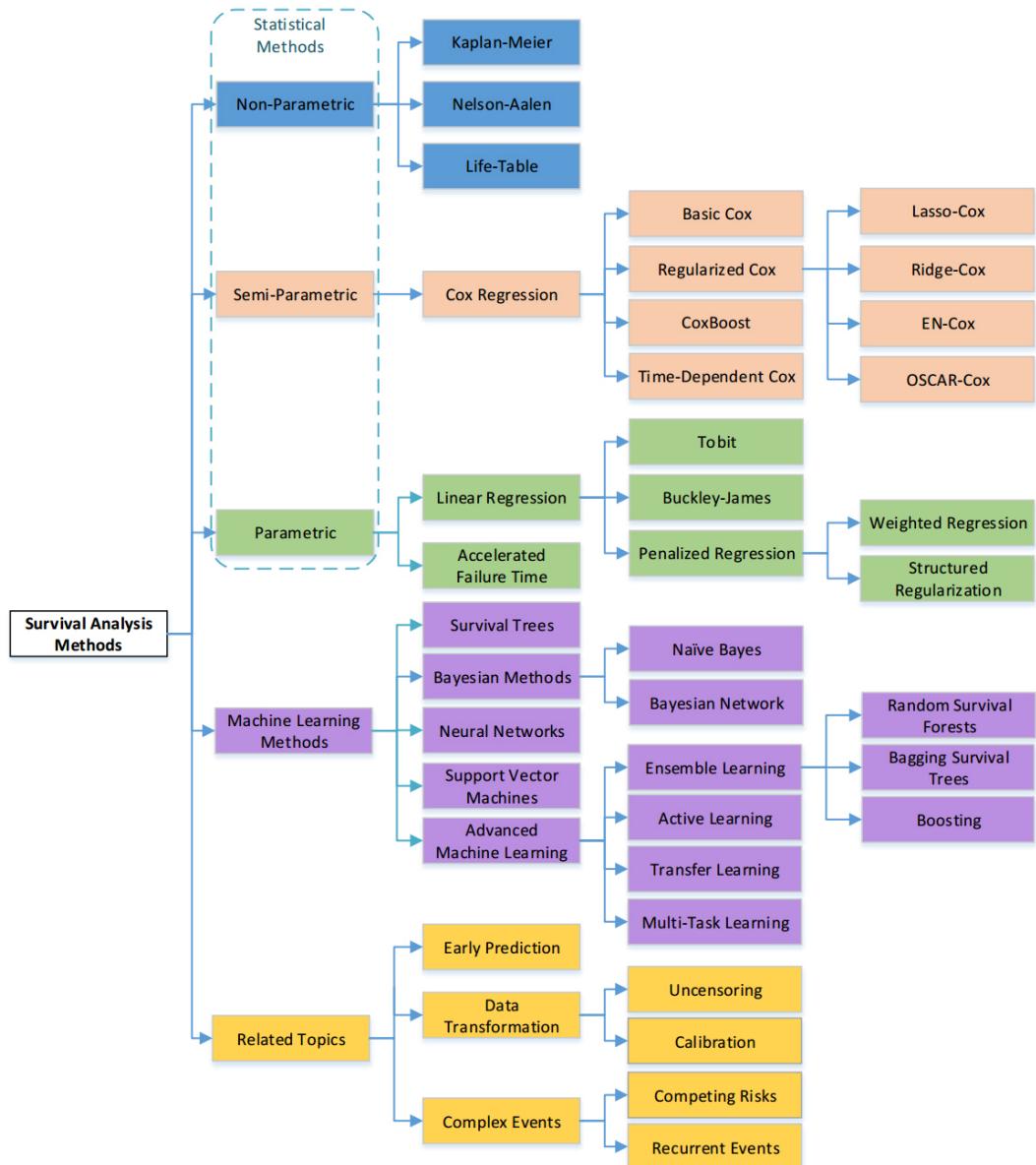


Figure 1: Ping Wang et al.에 의해 소개된 생존 분석 방법 분류

Ping Wang et al.[52]은 생존 분석 방법을 Figure 1과 같이 분류하였다.

분류의 깊이가 깊어질수록 분류 기준이 엄밀하지 않다는 점⁵은 어렵지만, 생존 분석에 대한 전체적인 큰 틀과 어떠한 방법들이 있는지를 확인할 수 있으며, CS 분야 연구자들에 의하여 나온 상당히 최근의 분류라는 점에 의의가 있다고 여겨진다.

⁵구체적으로는 다음과 같은 점을 들 수 있다.

- Naïve Bayes, Bayesian Network는 Bayesian Methods가 아니다.
- Advanced Machine Learning을 따로 분류하는 대신, Survival Tree와 Random Survival Forest, Bagging Survival Trees와의 관계를 보인다던지 하면 더 좋았을 것이다.
- Complex Event에 있어, Recurrent Event와 Competing Risk 뿐만 아니라, Clustered Failure time과 Multi-state Model도 중요한 이슈인데, 이에 대한 분류가 빠져있는 것이 아쉽다. 등.

Part III

생존 함수

4 반응변수

관심있는 반응변수를 T 라고 표시하자. (여기서 $T > 0$) (일반적인 통계 모형에서는 Y 로 주로 표시한다.)

T 는 0보다 큰 확률변수이며, 확률밀도함수(PDF) $f(t)$ 와 누적분포함수 $F(t)$ 를 가진다.

$f(t)$ 는 수명 분포의 의미를 가진다. 즉, 수명분포는 비음(non-negative)의 값만을 취하는 확률변수 T 의 분포로, 이 때, $t \leq 0$ 에 대하여 $F(t) = 0$ 을 만족한다. 아마도 우리 연구에서는 이를 연속형 확률 변수로 간주할 것이지만, 수명이 '복사기의 사용 횟수'와 같이 정의되는 경우, 이산형 확률 변수로도 간주할 수 있다.

시점 $t = 0$ 에서는 모든 시스템이 정상 가동 상태임을 가정하며, 따라서 불량품이나 가동불능 시스템은 처음부터 고려대상에서 제외된다.

수명은 비음의 값만을 취하기 때문에, 이 경우 $F(t)$ 는 다음과 같은 조건을 만족한다.

- $F(t) = P(T \leq t)$ 는 시스템이 시점 t 이전에 고장이 나는 확률이며, $\lim_{t \rightarrow \infty} F(t) = 1$ 을 만족한다.
- 모든 $t \leq 0$ 에 대하여 $F(t) = 0$. 즉, 음의 수명을 가질 수 없다.
- $t_1 < t_2$ 이면, $F(t_1) \leq F(t_2)$ 이다.

일반적인 통계분석에서는 관측치의 평균, 분산 등 요약통계량을 구하는 것이 의미있을 수 있지만, 생존분석에서는 censored data로 인해 이런 기술통계량이 그다지 의미있게 사용되지 못할 수 있다. 대신 직접적으로 분포함수를 추정함으로써 자료를 요약할 수 있다. 특히 평균 수명(mean life length)나 백분위수(percentile) 등은 그 자체로 수명분포를 완전하게 묘사할 수는 없으나, 수명에 관한 전체적인 경향을 설명할 수 있는 척도로 사용되고 있으며, 수명에 대한 정량적인 대표값이다.

우리가 알고자 하는 대상인 사건 발생 시간 $\tilde{T}_1, \dots, \tilde{T}_n$ 은 서로 독립이고, 동일한 분포 F 를 가진다.

5 생존 함수(survival function), 신뢰도 함수(reliability function)

사건까지의 시간(time-to-event)에 대한 현상을 설명하기 위한 생존함수(survival function)를 다음과 같이 정의한다.

$$S(t) = P(T > t) = \int_t^{\infty} f(x)dx = 1 - P(T \leq t) = 1 - F(t)$$

생존 분석에서는 $S(t)$, 신뢰성 공학에서는 $R(t)$ 로 표현한다. 하지만 이 둘의 차이는 없다. 이 보고서에서는 가급적 $S(t)$ 로 표기할 것이다.⁶ 혼동하지 않도록 주의하자.

- 생존함수는 비증가함수(non-increasing function)이다. 즉, $t_1 < t_2$ 이면, $S(t_1) \geq S(t_2)$.
- 사건 발생시간 t 는 처음부터 발생하거나 영원히 발생하지 않을 수 있으므로, 범위는 $t \in [0, \infty)$ 이다.
- $S(0) = 1$, $S(\infty) = \lim_{t \rightarrow \infty} S(t) = 0$
- 생존함수는 확률로 표현되므로 $0 \leq S(t) \leq 1$ 을 만족한다.
- $S(t) = P(T > t)$ 는 ”적어도 t 시점까지 생존할 확률”이며, 사건 발생이 t 시점 이후에 일어날 가능성이다. 신뢰성 공학이라면 ”나이가 t 인 시스템이 아직도 가동되고 있을 확률”이라고 표현할 것이다.
- $P(T \leq t)$ 는 t 시점 이전에 사건이 발생할 확률이다.
- $S(t)$ 에 대한 1차 도함수의 음의 값이, 확률밀도함수 $f(t)$ 가 된다. $f(t) = -\frac{dS(t)}{dt}$
- 일반적으로 $F(t) = 1 - S(t)$ 를 누적분포함수라고 하되, 신뢰성 공학에서는 신뢰도 함수와 대응하여 ”불신뢰도 함수(unreliability function)”이라고 부르기도 한다. 따라서, 신뢰도 함수와 불신뢰도 함수의 합은, 모든 t 에 대해서 1이다.

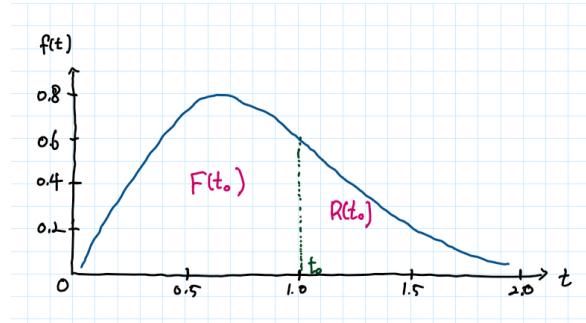


Figure 2: $F(t)$ 와 $R(t)$ 의 관계

Figure 3의 왼쪽은 연속변수에 대한 연속 생존함수를 보여주고, 오른쪽은 관측된 데이터에서 추정된 비모수적 계단함수 형태의 생존함수를 보여준다.

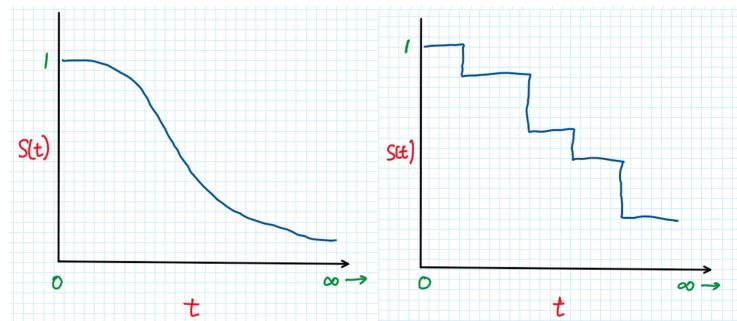


Figure 3: 이론적 생존함수(왼쪽)과 추정된 생존함수(오른쪽)

⁶ 다만 ”신뢰성 공학과 관련된 문맥에 따라” 혼용하여 사용할 수 있을 것이다.

6 위험 함수(hazard function), 위험률 함수(hazard rate function; 고장률 failure rate function)

위험률(hazard rate)은 ” t 시점에서 생존한 조건 하에서 사건이 발생할 확률”이다. 신뢰성 공학이라면, ”나이가 t 인 시스템이 가지는 고장 발생의 위험 정도”로 정의할 것이며, 이를 ”고장률(failure rate)”로 부르기도 한다.

Note 2. 위험률 함수 $h(t)$ 의 유도

시스템이 구간 $(t, \Delta t]$ 에 고장이 발생하는 확률은 다음과 같다.

$$\begin{aligned} P(t < T \leq t + \Delta t) &= \int_t^{t+\Delta t} f(u) du \\ &= F(t + \Delta t) - F(t) \end{aligned}$$

시스템이 시점 t 에 가동된다는 것이 주어지면,

$$\begin{aligned} P(t < T \leq t + \Delta t | T > t) &= \frac{P(t < T \leq t + \Delta t)}{P(T > t)} \\ &= \frac{F(t + \Delta t) - F(t)}{S(t)} \end{aligned}$$

이 값을 Δt 로 나누어서, 구간 $(t, \Delta t]$ 에서의 평균 조건부 확률을 구하면,

$$\frac{P(t < T \leq t + \Delta t | T > t)}{\Delta t} = \frac{F(t + \Delta t) - F(t)}{\Delta t \cdot S(t)}$$

위험률 함수 $h(t)$ 는 $\Delta \rightarrow 0$ 을 위한 평균 조건부 확률이다.

$$\begin{aligned} h(t) &= \lim_{\Delta t \rightarrow 0} \frac{F(t + \Delta t) - F(t)}{\Delta t \cdot S(t)} \\ &= \frac{1}{S(t)} \lim_{\Delta t \rightarrow 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \\ &= \frac{f(t)}{S(t)} \quad (t \geq 0) \end{aligned}$$

위험함수에 대한 해석을 생각해보자.

- 위험함수 값은 0 이상이다. $h(t) \geq 0, t \geq 0$
- $\int_0^\infty h(t) dt = \infty$
- 위험함수는 확률로 표현되므로, 일반적으로 관측 불가능한 양이다. 그러므로 데이터에 근거하여 추정해야 한다.
- 위험률은 표본 전체에 대한 특성이라기 보다는, 개인에 대한 특성으로 이해할 수 있다. 그러므로 개개인의 위험함수를 추정할 수 있다.
- 일반적인 위험함수의 세 가지 형태는 꾸준히 증가 / 꾸준히 감소 / 일정한 형태이다. (물론 다른 형태의 위험함수도 고려할 수 있다.)

7 누적 위험 함수(cumulative hazard function),
누적 고장률 함수(cumulative failure rate function)

누적 위험 함수는 시점 0부터 t 시점 까지의 누적된 위험률(누적된 고장률)로 정의한다. 수명 분포의 비모수적 분류에 유용하게 사용하는 척도이다.

$$\begin{aligned} H(t) &= \int_0^t h(u)du \\ &= \int_0^t \frac{\frac{d}{dt}[1 - S(t)]}{S(t)} \\ &= -\log S(t) \end{aligned}$$

Note 3. 누적함수를 이용하여 생존함수 구하기

생존함수를 누적함수를 이용해 다음과 같이 표현할 수 있다.

$$S(t) = \exp [-H(t)]$$

$$= \exp \left[\int_0^t h(u) du \right]$$

$H(t)$ 는 다음과 같은 성질을 가진다.

- $H(0) = 0$
 - $\lim_{t \rightarrow \infty} H(t) = \infty$
 - $H(t)$ 는 t 의 비 감소함수
 - $\frac{dH(t)}{dt} = h(t)$, 즉, 누적 위험 함수의 1차 도함수는 위험률 함수이다.

8 생존 함수간 관계

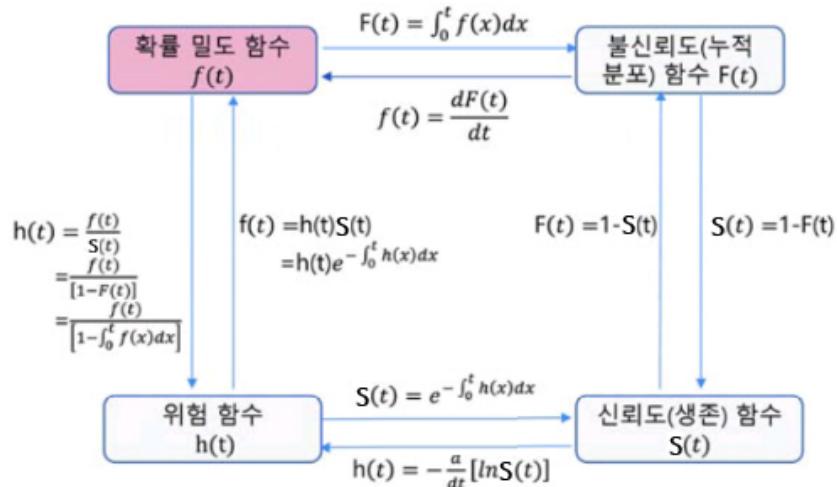


Figure 4: 생존 함수들 간 관계도

9 위험률 함수와 생존함수의 1:1 대응

위험률 함수(고장률 함수)가 중요한 척도 함수로 사용되고 있는 것은, 생존 함수(신퇴도 함수)와 1:1 대응 관계가 성립한다는 것이다. 이러한 성질은 위험률 함수에 대한 지식과 생존 함수에 대한 지식은 동치라는 것을 의미하며, 경우에 따라 생존 함수에 대한 직접적인 추정 대신, 위험률 함수에 대한 추정을 수행함으로써 수명 분포에 대한 분석을 할 수 있다는 의미가 된다.

위험률 함수가 주어지면, 생존 함수는 다음과 같이 계산된다.

$$\begin{aligned} h(t) &= \frac{1}{S(t)} \lim_{\Delta t \rightarrow 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \\ &= \frac{f(t)}{S(t)} = \frac{\frac{d}{dt} F(t)}{S(t)} = \frac{\frac{d}{dt} [1 - S(t)]}{S(t)} \\ &= \frac{-S'(t)}{S(t)} \\ &= -d \ln[S(t)] \end{aligned}$$

양변을 0부터 t 까지 적분하면, $S(0) = 1$ 이므로,

$$\begin{aligned} \int_0^t h(u) du &= \int_0^t \frac{-S'(u)}{S(u)} du \\ &= -\ln S(u)|_0^t \\ &= -\ln S(t) \end{aligned}$$

이 관계로부터, 다음과 같은 공식을 얻는다. 이를 역 공식(inversion formula)라고 부르며, 위험률 함수와 생존 함수의 1:1 관계를 나타낸다.

$$S(t) = e^{-\int_0^t h(u) du}$$

누적 위험 함수 $H(t)$ 를 이용하면, 이 역 공식은 다음과 같이 표시된다.

$$S(t) = e^{-H(t)}$$

9.1 역공식을 이용한 위험함수와 생존함수의 관계

여러 가지 형태의 위험함수에 대한 생존함수를 역 공식을 이용하여 구할 수 있다.

- 위험 함수가 상수인 경우 ($h(t) = \lambda$, $\lambda > 0$)

$$\int_0^t h(u) du = \int_0^t \lambda du = \lambda t$$

따라서, $S(t) = e^{-\lambda t}$ 가 얻어진다. 이 분포는 **지수분포**로 알려져 있다.

* 지수 분포에 대한 자세한 내용은 Chapter 14.11를 참고하자.

- 위험 함수가 선형 증가인 경우 ($h(t) = \alpha t + \beta$, $\alpha > 0, \beta > 0$)

$$\int_0^t h(u) du = \int_0^t (\alpha u + \beta) du = \frac{1}{2} \alpha t^2 + \beta t$$

따라서, $S(t) = e^{-\frac{1}{2} \alpha t^2 + \beta t}$ 가 얻어진다. 이 분포는 **선형증가 위험률 분포**로 알려져 있다.

* 선형 증가 분포에 대한 자세한 내용은 Chapter 14.21를 참고하자.

- 위험 함수가 멱함수인 경우 ($h(t) = \alpha t^{\beta-1}$, $\alpha > 0, \beta > 0$)

$$\int_0^t h(u) du = \int_0^t (\alpha u^{\beta-1}) du = \frac{\alpha}{\beta} t^\beta$$

따라서, $S(t) = e^{-\frac{\alpha}{\beta} t^\beta}$ 가 얻어진다. 이 분포는 **와이블 분포**로 알려져 있다.

- 위험 함수가 지수적으로 증가하는 경우 ($h(t) = \alpha e^{\beta t}$, $\alpha > 0, \beta > 0$)

$f(t) = \alpha e^{\beta t} e^{-\frac{\alpha}{\beta}(e^{\beta t}-1)}$, $S(t) = e^{-\frac{\alpha}{\beta}(e^{\beta t}-1)}$ 가 얻어진다.

이 분포는 극한값 분포(extreme value distribution)인 **Gompertz 분포**로 알려져 있다.

* Gompertz 분포에 대한 자세한 내용은 Chapter 14.16를 참고하자.

- 욕조 모양(bathtub-shaped)의 위험 함수인 경우

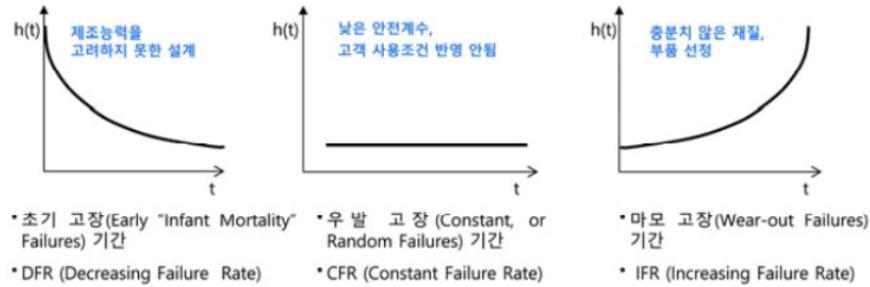


Figure 5: 일반적인 위험률 곡선의 유형

- **초기 고장기간(Infant Mortality Failures; DFR: Decreasing Failure Rate):**

일반적으로 대부분의 시스템에 대한 고장률의 함수는 초기의 짧은 기간동안 감소하게 된다.

부품의 결함이나 제조, 공정상의 실수, 설계상의 오류 등에 기인한 것이다. 의학에서는, 과거 의학이 발달하지 않은 시절에, 초기 신생아의 사망률은 높지만, 어느정도 나이가 지나면 사망률이 줄어드는 경우를 들 수 있다.

- **우발 고장기간(Constant Failures or Random Failures; CFR: Constant Failure Rate):**

초기 고장기간이 끝나게 되면, 일정한 고장률에 도달하게 된다.

이 일정기간 동안에는 고장이 우연히 발생하게 되며, 비교적 시스템의 나이에 영향을 받지 않는다.

- **마모 고장기간(Wear-out Failures; IFR: Increasing Failure Rate):**

시스템의 나이가 많아짐에 따라 고장률이 급속도로 증가하게 된다.

이 때는 부식, 산화, 마모, 피로도 누적, 균열, 수명이 짧은 부품의 사용, 충분치 않은 정비 등에 기인한다. 의학에서는, 나이가 들수록 사망할 확률이 증가하는 경우를 들 수 있다.

DFR, CFR, IFR을 모두 합친 뒤 매끄럽게 연결하면, 그림 6와 같은 욕조 곡선(Bathhub Curve)을 얻는다.

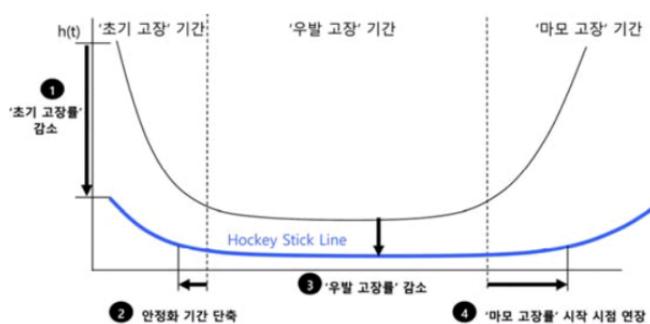


Figure 6: 욕조 곡선(bath-hub curve)

9.2 분포의 비모수적 분류

확률분포의 모수적 분류는 확률밀도함수나 누적분포함수의 형태에 의하여 이루어지는데 반해, 수명분포의 비모수적 분류는 수명분포를 특징지어주는 생존함수, 위험률함수, 평균잔여수명함수 등 여러 가지 척도함수들의 단조성(monotonicity)이나 기타 성질에 의하여 분류된다.

9.2.1 비모수적 군

Note 4. 평균 위험률(평균 고장률)

$$\bar{H}(t) = \frac{1}{t} H(t) = \frac{1}{t} \int_0^t h(u) du$$

- 위험률 함수 $h(t)$ 에 의한 분류:

만일 $h(t)$ 가 $t \geq 0$ 의 비감소함수이면, 수명분포 F 는 IFR(Increasing Failure Rate)에 속하고,

만일 $h(t)$ 가 $t \geq 0$ 의 비증가함수이면, 수명분포 F 는 DFR(Decreasing Failure Rate)에 속한다.

다만, 이 정의는 불신뢰도 함수 $F(t)$ 가 미분 불가능하면 적용되지 않는다. 따라서, 보다 일반적인 정의는 다음과 같다.

만일 $F(0) = 0$ 이고, 모든 $0 \leq s \leq \infty$ 와 $t > 0$ 에 대하여 $\frac{S(s+t)}{S(s)}$ 가 s 의 비증가(감소)함수이면, F 는 IFR(DFR)에 속한다.

- 누적 위험률 함수 $H(t)$ 에 의한 분류:

만일 $F(0) = 0$ 이고,

평균위험률 $\bar{H}(t)$ 이 $t \geq 0$ 의 비감소함수이면, 수명분포 F 는 IFRA(Increasing Failure Rate Average)에 속하고,

평균위험률 $\bar{H}(t)$ 이 $t \geq 0$ 의 비증가함수이면, 수명분포 F 는 DFRA(Decreasing Failure Rate Average)에 속한다.

다만, 이 정의는 불신뢰도 함수 $F(t)$ 가 미분 불가능하면 적용되지 않는다. 따라서, 보다 일반적인 정의는 다음과 같다.

만일 $F(0) = 0$ 이고, 모든 $-\frac{1}{t} \ln S(t)$ 가 비감소(증가)함수이면, F 는 IFRA(DFRA)에 속한다.

IFRA는 IFR보다 큰 군으로, 만일 수명분포가 IFRA에 속하는 독립적인 요소들로 구성된 대부분의 시스템의 수명도 IFRA에 속한다. 그러나 이러한 성질은 IFR 군에는 적용되지 않는다.

- 평균잔여수명함수 $m(t)$ 에 의한 분류:

- 만일 $m(t)$ 가 $t \geq 0$ 의 비증가함수이면, 수명분포 F 는 DMRL(Decreasing Mean Residual Life)에 속하고,
 $m(t)$ 가 $t \geq 0$ 의 비감소함수이면, 수명분포 F 는 IMRL(Increasing Mean Residual Life)에 속한다.

- $\mu = E(T)$ 가 존재한다고 하자. 만일 모든 $t \geq 0$ 에 대하여

$\int_t^\infty S(u)du \leq \mu S(t)$ 이면, F 는 NBUE(New Better than Used in Expectation)에 속하고,
 $\int_t^\infty S(u)du \geq \mu S(t)$ 이면, F 는 NWUE(New Worse than Used in Expectation)에 속한다.

만일 F 가 NBUE이면, 나이 t 의 중고시스템의 평균잔여수명이 새로운 시스템의 평균수명보다 적다는 것을 의미한다.

- $\mu = E(T)$ 가 존재한다고 하자. 만일 모든 $t \geq 0$ 에 대하여

$\int_t^\infty S(u)du \leq \mu e^{-\frac{t}{\mu}}$ 이면, F 는 HNBUE(Harmonic New Better than Used in Expectation)에 속하고,
 $\int_t^\infty S(u)du \geq \mu e^{-\frac{t}{\mu}}$ 이면, F 는 HNWUE(Harmonic New Worse than Used in Expectation)에 속한다.

- 생존 함수 $S(t)$ 에 의한 분류:

- 모든 $s \geq 0$ 과 $t \geq 0$ 에 대하여,

$S(s+t) \leq S(s)S(t)$ 이면, F 는 NBU(New Better than Used)에 속하며,
 $S(s+t) \geq S(s)S(t)$ 이면, F 는 NWU(New Worses than Used)에 속한다.

만일 F 가 NBU이면, 중고시스템의 신뢰도는, 나이에 관계 없이 새로운 시스템의 신뢰도에 비하여 적다.

- 고정된 $t_0 \geq 0$ 과 모든 $t \geq 0$ 에 대하여,

$S(t+t_0) \leq S(t)S(t_0)$ 이면, F 는 NBU- t_0 (New Better than Used of age t_0)에 속하며,
 $S(t+t_0) \geq S(t)S(t_0)$ 이면, F 는 NWU- t_0 (New Worse than Used of age t_0)에 속한다.

만일 F 가 NBU- t_0 이면, 나이 t_0 의 중고시스템의 신뢰도는, 새로운 시스템의 신뢰도에 비하여 적다는 것을 의미한다.

9.2.2 비모수적 군들의 상호관계

앞서 정의된 수명 분포들의 비모수적 군은, 상호 간에 다음과 같은 관계를 갖고 있다.

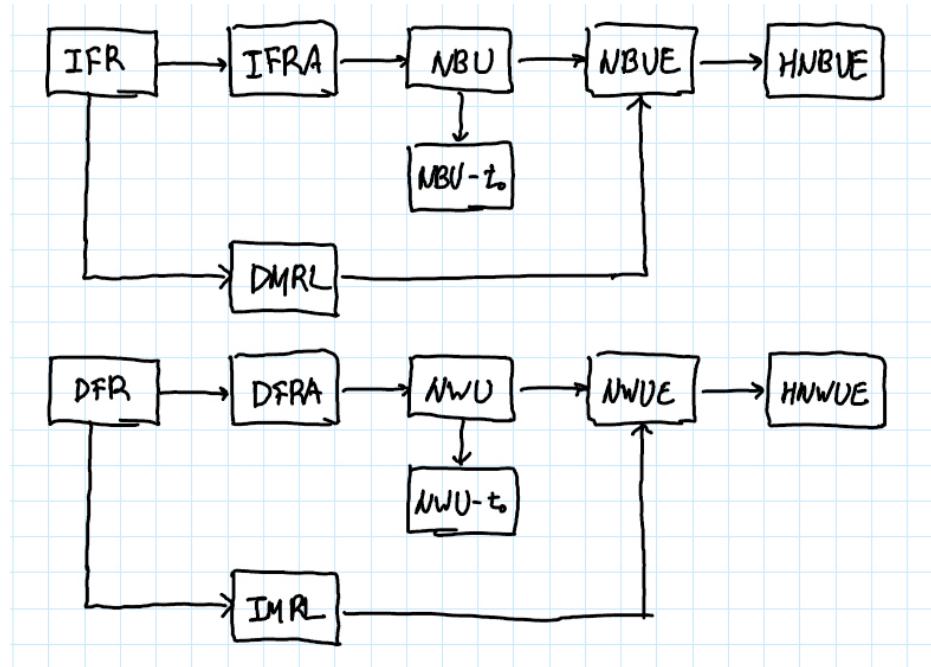


Figure 7: 비모수적 군들의 상호 관계

Part IV

척도 함수

생존률, 신뢰성, 위험률 등에 대한 정량적인 평가는 다음 2가지 견해가 있다.

- 재발(고장) 발생까지 시간 길이의 길고 짧음

재발(고장) 발생까지의 시간 길이에 착안해서, 그 시간이 길면 제품의 신뢰성이 높다고 생각하는 견해이다.

이 경우 생존자료 척도로서 재발(고장)까지의 시간을 확률변수 T 로 표시한다면, 생존함수(신뢰도 함수), 평균고장시간(MTTF; mean time to failure), 백분위수(percentile), Bearing Life 등을 정의한다.

- 일정시간 내 재발(고장) 발생 수의 많고 적음

재발(고장) 수의 많고 적음에 착안하여, 재발(고장)이 적을수록 생존률(신뢰성)이 높다고 하는 견해이다.

이에 대한 대표적인 척도로, 위험률(고장률)에 대해, 고장(재발) 난 경우 수리(치료)를 생각하는 수리계(repairable system)과 수리(치료)를 생각하지 않는 비수리계(non-repairable system)로, 이 둘의 정의가 조금 다르다.

10 '고장(재발) 발생까지 시간 길이의 길고 짧음'에 따른 척도 함수

10.1 생존 함수(Survival Function)

Chapter 5를 참고하자.

10.2 평균 수명(mean time to failure)

평균수명(mean time)은 시스템이 고장 날 때까지의 평균 시간으로, 수명 T 의 기대값으로 정의된다.⁷

시스템의 평균 수명은 시스템의 수명을 대표하는 가장 중요한 척도로서, 분포의 무게중심에 관한 정량적인 값을 나타내며, 항상 양의 값을 취한다.

앞서 언급했듯이, 평균수명은 평균잔여수명함수(mean residual life function)에서 $t = 0$ 인 경우로, $m(0)$ 이다.

일반적으로 평균 수명은 T 의 확률밀도함수를 이용하여 다음과 같이 계산된다.

$$E[T] = \int_0^\infty tf(t)dt$$

Note 5. $T \geq 0$ 인 경우 평균수명의 유도

시스템의 수명을 나타내는 확률변수 T 는 항상 비음(non-negative)의 값을 취하므로, 만약 $E(T)$ 가 존재한다면, 즉 $E(T) < \infty$ 이면, $E(T)$ 를 Note 5와 같이 보다 편리하게 유도할 수 있다.

$$\begin{aligned} E[T] &= \int_0^\infty tf(t)dt \\ &= \int_0^\infty td[-S(t)] \\ &= t[-S(t)]_0^\infty - \int_0^\infty [-S(t)] dt \\ &= -\lim_{y \rightarrow \infty} yS(y) + \int_0^\infty [S(t)] dt \end{aligned}$$

이제, $E(T) < \infty$ 이므로, $yS(t) = y \int_y^\infty f(t)dt < \int_0^\infty tf(t)dt$ 성립되며, $\lim_{y \rightarrow \infty} \int_0^\infty tf(t)dt = 0$ 이므로, $\lim_{y \rightarrow \infty} yS(t) = 0$ 된다.

따라서, 확률변수 $T \geq 0$ 인 경우, 다음과 같은 결론을 얻을 수 있다.

$$E(T) = \int_0^\infty S(t)dt$$

⁷ 신뢰성 공학에서는 이 평균수명에 대해, 비수리 시스템에서는 MTTF(Mean Time to Failure), 수리 시스템의 경우에는 수리에 의해 완전히 복구할 수 있다는 가정 하에 MTBF(Mean Time Between Failure)라고 부르기도 한다.

10.3 평균 고장 간격(mean time between failure)

식 (1)이 일정한 경우, $h(t) = \lambda$ 와 같이 표현할 수 있다.

이의 역수를 평균 고장 간격(MTBF; mean time between failure)⁸이라고 한다.

$$MTBF = \frac{1}{\lambda} = \frac{1}{고장률} = \frac{\text{일정 시간}}{\text{일정 시간 내의 고장 수}}$$

이는 수리 후 다음 고장이 발생할 때 까지의 시간(고장 간격)의 평균의 의미를 지닌다.

시스템이 비수리계(non-repairable system)인 경우의 평균 수명을 MTTF(mean time to failure)⁸,

시스템이 수리계(repairable system)인 경우의 평균 수명을 MTBF로 표시한다.

다만, 이 둘은 혼용되기도 한다.

위험률(고장률)이 시간당 λ 라면, 이 장비의 평균 수명을 다음과 같이 구할 수 있다.

$$MTBF = MTTF = E[T] = \int_0^\infty S(t)dt = \int_0^\infty e^{-\lambda t} dt = \frac{1}{\lambda}$$

10.4 백분위수

시스템의 수명 분포가 알려지게 되면, 적률(moment)이나 백분위수(percentile)에 대한 계산이 가능해진다. 이러한 척도는 수명분포 자체보다는 수명에 대해 적은 정보를 제공하지만, 수명분포의 전체적인 패턴을 요약하는 유익한 대표값 중 하나이다.

수명분포와 함께 수명분포의 중심을 나타내는 또 다른 정량적인 척도로 중앙값(median)과 최빈값(mode)이 있다.

이 중 최빈값(mode)는 확률밀도함수가 최대값을 취하는 시점을 나타내며, 중앙값(median)은 확률밀도함수의 면적을 정확하게 양분하는 시점을 나타낸다. 즉, $S(t) = 0.5$ 를 만족하는 t 의 분포의 중앙값이다.

확률밀도함수, 생존함수(신뢰도함수) 및 불신뢰도함수를 각각 $f(t)$, $S(t)$, $F(t)$ 로 각각 표시하면, 100p 백분위수(100pth percentile, pth fractile)은 다음의 두 식을 만족하는 t 값으로 정의되며, t_p 로 나타낸다. (단, $0 < p < 1$)

$$P(T \leq t) \geq p$$

$$P(T \geq t) \geq 1 - p$$

만일 T 가 연속확률변수이면, 위 두 식은 $p \leq F(t) \leq p + P(T = t) = p^o$ 되며,
따라서, $F(t_p) = p$ 또는 $S(t_p) = 1 - p$ 가 된다.

10.5 Bearing Life

생존률(신뢰성)의 평가는, 고객에 대한 제품의 보증을 중요하게 생각하는 입장에서 보면, 평균에 대한 개념 보다는, '제품을 언제까지 안심하고 사용할 수 있는가'와 같이 제품 하나하나에 대해 수명을 보증하는 "한계 보증치"가 중요하다.

Bearing Life는 산포의 크기를 고려한 한계치 보증의 입장에서 고려되고 있는 척도이다.

B_{10} 수명(Bearing 10 % life)은, 전체의 10 %가 고장(재발)날 때까지의 시간을 의미한다.

$$\int_0^{B_{10}} f(t)dt = 0.10$$

생존 함수와 B_{10} 수명과의 관계는 다음과 같다.

$$S(t = B_{10}) = \int_{B_{10}}^\infty f(t)dt = 0.90$$

⁸Chapter 10.2 참고

11 '고장(재발) 발생까지 시간 길이의 길고 짧음'에 따른 척도 함수

11.1 평균 잔여 수명 함수

평균잔여수명함수(mean residual life function)는, 시스템이 나이 t 까지 가동된다는 것이 주어져 있을 때, 앞으로의 잔여 수명에 대한 기대값으로 정의된다.

$$m(t) = E[T - t | T > t]$$

평균수명이 유한한 값을 가지는 경우, 평균잔여수명함수 $m(t)$ 는 모든 $t \geq 0$ 에 대해 다음과 같은 성질을 가진다.

- $m(t) \geq 0$
- $m'(t) \geq -1$
- $\int_0^\infty \frac{1}{m(t)} dt = \infty$
- 새로운 시스템의 평균수명⁹은 $t = 0$ 에서의 평균잔여수명이다.

Note 6. 평균잔여수명함수의 유도

먼저 $T > t$ 가 주어졌을 때, 다음과 같은 T 의 조건부 확률밀도함수가 필요하다.

$$f_{T|T>t}(u) = \frac{f(u)}{S(u)}, \quad u > t$$

이를 바탕으로, 조건부 기대값은 다음과 같이 계산된다.

$$\begin{aligned} m(t) &= E[T - t | T > t] \\ &= \int_t^\infty (u - t) f_{T|T>t}(u) du \\ &= \int_t^\infty (u - t) \frac{f(u)}{S(u)} du \\ &= \frac{1}{S(u)} \int_t^\infty (u - t) f(u) du - t \\ &= \frac{1}{S(u)} \int_0^\infty S(u + t) du \end{aligned}$$

Note 7. 평균잔여수명함수와 위험률 함수, 생존 함수간의 1:1 관계

$$\begin{aligned} m'(t) &= \frac{S(t) \frac{d}{dt} \int_t^\infty S(u) du - \int_t^\infty S(u) du \frac{d}{dt} S(t)}{[S(t)]^2} \\ &= \frac{-[S(t)]^2 + f(t) \int_t^\infty S(u) du}{[S(t)]^2} \\ &= -1 + \frac{f(t) \int_t^\infty S(u) du}{[S(t)]^2} \\ \therefore h(t) &= \frac{f(t)}{S(t)} = \frac{1 + m'(t)}{m(t)} \end{aligned}$$

따라서, $m(t)$ 과 $h(t)$ 는 1:1 관계이다. 또한, $S(t)$ 와 $h(t)$ 가 1:1 관계이기 때문에, $m(t)$ 과 $S(t)$ 도 1:1 관계이다.

⁹"평균수명(mean time)"과 "평균잔여수명(mean residual life)"은 구분되는 개념이다.

11.2 가용도 함수

수리가 가능한 시스템의 경우, 현실적으로 수리나 보전에 일정한 시간이 걸리는 경우가 대부분이므로, ”수리 시간”을 고장 시간과 별도로 모형화하는 것이 보다 합리적일 수 있다.¹⁰ 따라서, 이러한 경우에는 시점 t 에서 시스템이 실제로 가동 상태에 있는지에 대하여 관심을 가지게 된다.

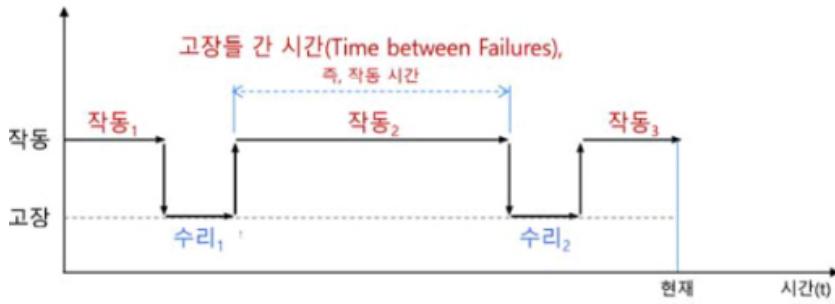


Figure 8: 수리가 가능한 시스템

시점 t 에서 수리 가능 시스템의 상태 함수를 다음과 같이 정의한다.

$$X(t) = \begin{cases} 0 & \text{시스템이 시점 } t \text{에서 고장 상태} \\ 1 & \text{시스템이 시점 } t \text{에서 가동 상태} \end{cases}$$

수리 가능 시스템의 노화에 의한 성능 평가를 위한 척도 중 하나가 **가용도**로, 주어진 시점 t 에서 시스템이 가동 상태에 있는 확률로 정의된다.

가용도를 측정하는 척도에는 다음과 같은 것이 있다. (단, $A(t)$ 는 시스템이 시점 t 에서 가동되고 있는 확률)

- 가용도 함수

$$A(t) = P[X(t) = 1] = E[X(t)] \quad (t > 0)$$

- 극한 가용도

$$A(t) = \lim_{t \rightarrow \infty} A(t)$$

만일, 시스템의 고장률(위험율)과 수리율이 각각 상수인 λ 와 μ 이면, 극한 가용도 $A = \frac{\mu}{\lambda + \mu}$

만일 어느 시스템에 대하여 $A = 0.96$ 이면, 장기적으로 그 시스템은 전체 기간의 96% 기간 동안 가동되고 있다는 것을 의미한다.

수리 불가능 시스템에서는 $A(t) = S(t)$ 가 된다.

¹⁰ 의학에서는, 재발 가능한 사건(recurrence available event)에 있어, ”치료 기간”을 별도로 모델링하는 경우를 들 수 있다.

11.3 위험률(고장률)

11.3.1 비수리계(non-repairable system)의 위험률(고장률)

비수리계¹¹에서의 위험률(고장률)이란, 임의의 시점 t 까지 동작을 계속해 온 시스템 또는 기기가 그 시점에서 계속해서 단위 시간당 고장이 나는 확률을 말한다.

이를 확률변수 T 의 조건부 확률을 이용하여 표시하면 다음과 같다.

$$P[t < T \leq t + dt | T > t]$$

이는 즉, 시점 t 에서 고장나지 않은 아이템이, 다음 작은 시간 $(t, t + dt)$ 사이에서 고장이 날 확률을 의미한다.

'단위 시간당 고장 확률'로 변환하면 다음과 같고, 이를 작은 시간 $(t, t + dt)$ 에 있어서의 평균 고장률(average failure rate)이라고 한다. 자세한 것은 Chapter 11.3.3을 참고하도록 하자.

$$P\left[\frac{t < T \leq t + dt | T > t}{dt}\right]$$

11.3.2 수리계(repairable system)의 위험률(고장률)

수리계¹²에 대해 시간 $(0, t)$ 의 고장 수(재발 수)를 $n(t)$ ¹³라 두면, 이것이 확률변수가 된다.¹⁴ 이 확률변수에 대해 단위 시간당 고장 발생 확률을 다음과 같이 생각할 수 있다.

$$\text{위험률(고장률)} = \lim_{dt \rightarrow 0} \frac{1}{dt} P[(n(t + dt) - n(t)) \geq 1] = \frac{\text{일정 시간 내의 고장 수}}{\text{일정 시간}} \quad (1)$$

11.3.3 평균 위험률(평균 고장률, average failure rate)

구간 $[0, t]$ 의 누적 위험률(누적 고장률)을 $H(t)$ 라 하면, $S(t) = e^{-\int_0^t h(t)dt}$ 로부터,

$$H(t) = \int_0^t h(t)dt = -\ln S(t)$$

구간 $[t_1, t_2]$ 의 구간 위험률(구간 고장률) $FR(t_1, t_2)$ 은

$$FR(t_1, t_2) = H(t_2) - H(t_1) = \ln S(t_1) - \ln S(t_2)$$

구간 $[t_1, t_2]$ 의 구간 평균 위험률(구간 평균 고장률) $AFR(t_1, t_2)$ 은

$$AFR(t_1, t_2) = \frac{\ln S(t_1) - \ln S(t_2)}{t_2 - t_1}$$

구간 $[0, t]$ 의 평균 위험률(평균 고장률) $AFR(t)$ 은

$$AFR(t) = -\frac{\ln S(t)}{t}$$

¹¹고장이 났을 때 수리를 생각하지 않는 경우

¹²아이템의 고장 시, 수리에 의해 그 가능을 회복하는 경우

¹³수리 시간은 무시하였다. 수리 시간을 고려한다면 별도의 모델링이 필요하다.

¹⁴정확히는, 시간 t 를 파라미터로 하는 확률변수이므로, 확률 과정(random process)가 된다.

Part V

수명 자료(Life Data), 중도 절단 자료(Censored Data)

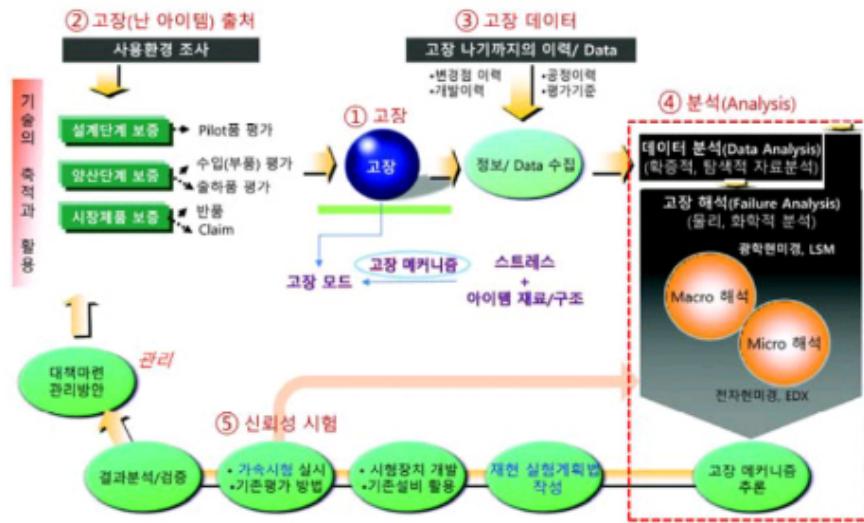


Figure 9: 신뢰성 보증 사이클

"고장이 일어나는 유형", 즉 "위험 유형"은 앞서 욕조 곡선(Bath-tub Curve)를 이용하여 소개한 바 있다. 도메인에 따라 "위험"이 일어나는 매커니즘은 달라질 것이다. 신뢰성 공학에서는 기계의 고장(failure)이 "설계 단계 보증", "시장 제품 보증"의 업무 결과 나타난다고 설명하고 있으며, 고장의 원인을 Figure 10[3]와 같이 정리하고 있다.

유형(Type)	기계적 (Mechanical)	열적 (Thermal)	화학적 (Chemical)	전기적 (Electrical)
부하(Load) - 시간 의존성 외력	<ul style="list-style-type: none"> 외력(External Forces) 기계적 충격(Mechanical Shock) 온도 충격(Temperature Shock) 온도 구배(Temperature Gradient) 진동(Vibration) 	<ul style="list-style-type: none"> 온도 온도 충격(Thermal Shock) 온도 구배(Temperature Gradient) 	<ul style="list-style-type: none"> 습도(Humidity) 화학 악품 노출(Chemical Exposure) 방사선(Radiation) 	<ul style="list-style-type: none"> 전압(Voltage) 전류(Current)
용이 사용 예	<p>② (부하) 기계적 - 외력 ① 고장 위험 ③ Stress 유발됨 시간 경과 ⑤ (고장) 모드 종 (고장) 파손</p> <p>④ (고장 메커니즘) 기계적 - 피로</p>			
고장 메커니즘(Failure Mechanism)	<p>파로 (Fatigue):</p> <ul style="list-style-type: none"> 크리프(Creep) 마모(Wear) 응력 공극(Stress Voids) 	<ul style="list-style-type: none"> 확산 관련 금속간 성장(Diffusion Related Intermetallic Growth) 힐록 형성(Hilllock Formation) 	<ul style="list-style-type: none"> 부식(Corrosion) 음력 부식(Stress Corrosion) 덴드라이트 성장(Dendrite Growth) 해중합(De-polymerization) 	<ul style="list-style-type: none"> 전기적 이동(Electro Migration) 시간 의존 절연 파괴(TIDB, Time-Dependent Dielectric Breakdown)
고장 메커니즘 (Failure Mechanism)	<p>과 응력 (Overstress):</p> <ul style="list-style-type: none"> 과도한 변형(Excess Deformation) 황복(Yield) 균열(Fracture) 	<ul style="list-style-type: none"> 열 과응력(Thermal Overstress) 융해(Melting) 	=	<ul style="list-style-type: none"> EOS(Electrical Overstress) ESD(Electrostatic Discharge) 래치 업(Latch-up)

Figure 10: "고장"이 일어나는 유형 및 매커니즘

이러한 수명 데이터(Life Data)의 유형은 다음과 같다.

- 완전 자료(complete data): 각 아이템의 고장(재발) 발생 시간을 정확하게 알고 있는 데이터
- 중도 절단 자료(censored data)

- 우측 중도절단 자료(right-censored data): 수명 시험 동안 고장(재발)이 발생하지 않은 데이터
- 좌측 중도절단 자료(left-censored data): 특정 시간 전에 고장(재발)이 발생한 데이터
- 구간 중도절단 자료(interval-censored data): 특정 시간 사이에 고장(재발)이 발생한 데이터

중도절단의 가장 일반적인 형태는 **구간중도절단(interval censoring)**이다. 중도절단의 가장 일반적인 개념으로 우중도절단(right censoring)과 좌중도절단(left censoring)은 구간중도절단의 특별한 형태이다.

구간중도절단자료란 \tilde{T}_i 를 정확하게 관찰할 수 없으며, 다만 $(L_i, R_i]$ 라는 구간 안에 발생하였다는 것을 알 수 있다.

$$\tilde{T}_i \in (L_i, R_i] \quad (\text{단}, L_i \leq R_i)$$

여기에서, 만일 한쪽 방향의 중도절단만을 가정한다면 이것이 우중도절단(right censoring) 혹은 좌중도절단(left censoring)이 된다.

중도절단시간 C_1, \dots, C_n 이 서로 독립이고, 동일한 분포 F_c 를 가진다고 하자.
만약

- $\tilde{T}_i \leq C_i$ 이면 우중도절단(right censored)
- $\tilde{T}_i \geq C_i$ 이면 좌중도절단(left censored)

이 된다. 좌중도절단이 일어난 경우, 정확한 사건발생시간을 알 수 없다.

그리고 학교 수업시간이나 텍스트북 등으로 배우는 것은 대부분 우중도절단(right censored) 자료들이었을 것이다.

사실 구간중도절단자료의 발생빈도가 매우 높긴 하지만 계산의 복잡성 등으로 인해 관련 연구가 부족한 현실이다. 구간중도절단이 우중도절단의 generalized 형태라고 할 수 있기는 하지만, 중도절단 자료의 특성상 우중도절단 자료를 분석하는 방법으로 구간중도절단 자료를 분석하는 데에 사용하는 것은 곤란하다는 점만 알아두자.

중도절단의 이유는 다음 3가지로 분류할 수 있다.

- **연구 종료:** 연구가 종료될 때 까지 사건이 일어나지 않았다. 이를 행정상 중도절단(administrative censoring)이라고도 한다. 이 경우 중도절단은 관심있는 사건과 무관하다고 할 수 있다.
- **추적 실패(Loss to follow up):** 연구자 연락을 두절한 경우, 더 이상 추적 조사가 불가능하게 된다. 이 경우, 두절된 원인을 조사하는 것이 필요하다.
 - 임상연구에서 참여자가 너무 건강하여 치료법에 대한 필요를 느끼지 못해 연구에서 중도 탈락할 경우, 추정된 생존률은 실제보다 더 낮을 수 있다.
 - 약물의 부작용 등으로 연구에서 일찍 탈락한 겸응, 더 높은 생존률로 잘못 추정될 수 있다.
- **경쟁위험모형(competing risk):** 다른 종류의 사건 발생으로 인해 관심있는 사건이 중도절단된 경우에, 경쟁위험모형을 고려해야 한다.
- **측정 기구의 한계:** 가령 100kg까지의 무게를 챌 수 있는 경우, 120kg인 개인의 몸무게는 100+로 기록된다.

12 우중도절단(right censoring)

Figure 11를 살펴보자.

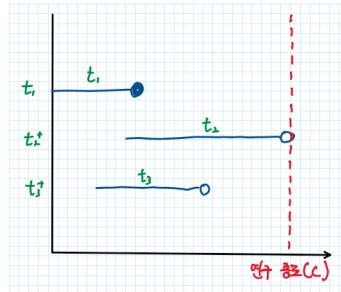


Figure 11: 우중도절단(right-censored)

- 첫 번째 관측 대상은, $t = 0$, 즉 연구 시작 시점부터 시작하여, t_i 시점에서 사건 발생을 겪었다.
- 두 번째 관측 대상은, 연구 시작 후 어느정도 시간이 지난 후에 연구에 참여하여 연구가 종료할 때 까지 사건을 경험하지 않았다.
- 세 번째 관측 대상은, 연구 시작 후 어느정도 시간이 지난 후에 연구에 참여하였으나, 어떠한 이유로 하여금 도중에 중도 절단하게 되었다.

사건 발생 시간 $\tilde{T}_1, \dots, \tilde{T}_n$ 은 서로 독립이고, 동일한 분포 F 를 가지며, 이는 우리가 알고자 하는 대상이다. 한편, 우중도절단시간 C_1, \dots, C_n 또한 서로 독립이고, 동일한 분포 F_C 를 가진다고 하자. 만약 $\tilde{T}_i \leq C_i$ 이면, T_i 를 관측할 수 있지만, 그렇지 못한 경우 C_i 가 관측된다. 이를 우중도절단(right censoring) 되었다고 한다.

앞으로 특별한 언급이 없으면, 이 노트에서 이야기하는 중도절단은 우중도절단(right censoring)을 말한다.

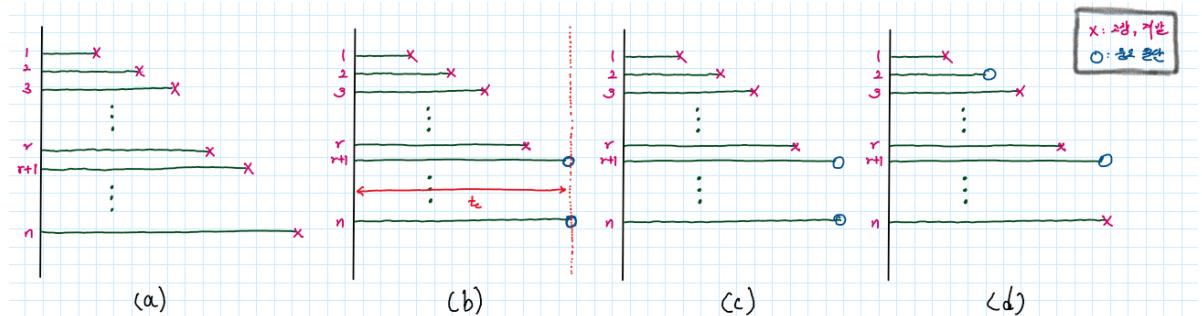


Figure 12: (a): 완전 자료(complete data), (b): Type 1 우중도절단, (c): Type 2 우중도절단, (d): 임의 우중도절단

일반적인 우중도절단은 다음의 3가지 유형으로 나눌 수 있다.

- Type 1 우중도절단(정시중단자료):** 모든 실험 object들은 미리 정해진 시점, 가령 C 까지 관측된다. 즉, 모든 실험 object의 우중도절단 시간이 동일하다. ($C_1 = C_2 = \dots = C_n = C$) 따라서, 중도절단시점 전에 사건이 발생되면, 그들의 T_i 를 관측할 수 있고, 그렇지 않은 경우 $C_i = C$ 가 사용된다.
- Type 2 우중도절단(정수중단자료):** 전체 실험 object 중 미리 정해놓은 사건 발생률을 가질 경우(예를 들어, 전체 실험 object 중 70%가 사건을 가질 경우) 관측을 중지한다고 하자. 따라서 언제 관측이 종료될지는 알 수 없다. 이 경우에도, (비록 알려져 있지 않지만) 모든 관측 object는 같은 중도절단시점 $C_i = C$ 를 가지게 된다.
- 임의(random) 우중도절단:** 위 2가지 유형은, 모든 object의 연구시작 시점이 동일하다는 가정이 있다. 하지만 실제 자료에서는 서로 다른 시점에서 연구에 참여할 수 있으며, 연구가 종료되지 않았음에도 불구하고 여러가지 이유로 더이상 연구에 참여하지 못할 수도 있다.

우리가 사용할 수 있는 자료는, 중도절단되지 않은 실험 대상자의 정확한 사건발생시점(\tilde{T}_i)과, 중도절단된 대상자에 대한 관측종단시점(C_i)이 된다. 즉, 중도절단된 대상자에 대해선, 중단시점까지는 사건이 발생되지 않음을 알 수 있으므로, 중도절단 시점이 사용된다. 이러한 자료는 다음과 같이 표현한다.

$$\{(T_i, \delta_i), i = 1, \dots, n\} \quad (2)$$

12.1 완전 자료(complete data)

시험 대상이 되는 n 개의 표본을 가지고, 시점 $t = 0$ 에서 동시에 수명 시험을 시작한다고 하자. 만일 모든 표본의 고장(재발) 시간 t_1, t_2, \dots, t_n 이 관측되었다면 완전자료가 얻어지게 되며, 이 경우 수명 시험에 소요되는 시간은 $t_{(n)} = \max(t_1, t_2, \dots, t_n)$ 이다.

만일 수명분포에 대한 형태를 모르는 경우에는, 비모수적 방법에 의하여 수명분포의 생존함수 $S(t)$ 를 추정하여야 하는데, 완전자료를 이용하는 경우에 보편적으로 경험적 추정 방법을 사용한다.

이 방법에서는 각각의 고장시간 $t_i (i = 1, 2, \dots, n)$ 에서, 생존율(신뢰도)가 $\frac{1}{n}$ 만큼식 감소하는 계단함수의 형태로 생존함수를 추정한다. 즉, 시점 t 에서의 생존함수에 대한 비모수적 추정값은 다음과 같이 계산된다.

$$\hat{S}_n(t) = \frac{\text{시점 } t\text{에서 아직도 가동되고 있는 부품 수(아직 재발하지 않은 사람 수)}}{n} \quad t \geq 0$$

따라서, 경험적 추정방법에 의한 $F(t)$ 의 추정값은 $\hat{F}_n(t) = 1 - \hat{S}_n(t)$ 에 의해 얻어지며, $\hat{F}_n(t)$ 을 경험적 분포 함수(empirical distribution function)이라고 부른다. n 을 무한히 크게 하면, 모든 t 에 대하여 $\hat{S}_n(t)$ 와 $\hat{F}_n(t)$ 가, 참값 $S(t)$, $F(t)$ 에 각각 수렴한다.

경험적 분포함수를 구하기 위하여, 먼저 n 개의 완전자료 t_1, t_2, \dots, t_n 을 작은 것부터 큰 순으로 나열하여, $t_{(1)} < t_{(2)} < \dots < t_{(n)}$ (순서 통계량으로 정의)으로 표시하자. 그러면, $t_{(i)} \leq t \leq t_{(i-1)}$, ($i = 0, 1, \dots, n$ 에서), 다음과 같이 추정된다.

$$\hat{F}_n(t) = \frac{i}{n} = \frac{\text{시점 } t\text{까지의 고장 부품 수(재발 환자 수)}}{\text{총 표본 수}}$$

여기에서, $t_{(0)} = 0$ 과 $t_{(n+1)} = \infty$ 로 정의한다. 따라서,

$$\hat{S}_n(t) = 1 - \hat{F}_n(t) = \frac{n-i}{n} = \frac{\text{시점 } t\text{에서 아직도 가동되고 있는 부품 수(아직 재발하지 않은 환자 수)}}{\text{총 표본 수}}$$

12.2 Type 1 우중도절단 자료(정시중단자료)

완전자료에서와 마찬가지로 수명 시험은 시험 대상이 되는 n 개의 표본을 가지고 시점 $t = 0$ 에서 동시에 시작한다. 이 경우에는 n 개의 표본에 대한 고장(재발) 시간이 모두 관측될 때까지 기다리지 않고, 미리 정해진 시점 t_c 에서 시험을 중단한다.

이러한 시험 중단은 시간적인 제약이나 비용 등의 이유로, 모든 표본이 고장 날 때까지(재발 할 때까지) 기다리는 것이 불가능한 경우에 적용되는 것으로, 특히 높은 신뢰도를 가진 전자부품 등의 경우에는 모든 부품에 대한 고장 자료를 얻는 것이 매우 어렵다.

따라서, 표본에 대한 시험기간 $(0, t_c)$ 동안에는, 고장 날 일부 표본의 고장시간만이 관측되고, 나머지 표본에 대해서는 고장 시간이 t_c 를 초과한다는 정보만 얻은 채, 정확한 고장 시간을 관측하는 것이 불가능하게 된다. 이러한 경우에 얻어지는 자료가 **Type 1 우중도절단 자료(정시중단자료)**이며, 만일 모든 표본이 시험기간 동안에 고장나는 경우에는 완전자료가 얻어지게 된다.

정시 중단의 경우에는, 시험 중단 시점이 미리 정해지기 때문에, 시험기간 $(0, t_c)$ 동안에 발생하는 고장(재발) 수는 이산화률 변수¹⁵가 되며, 표본 수가 많아지면 고장(재발) 수 또한 증가하게 된다. 시험을 시작한 n 개의 표본에서 얻어지는 정시중단자료의 예는 다음과 같다.

$$t_1, \dots, t_r, t_c^+, \dots, t_n^+$$

여기에서 $r (\leq n)$ 개의 고장 시간은 t_1, \dots, t_r 로 표시되고, 나머지 $(n-r)$ 개의 관측되지 않은 고장시간은 t_c^+ 로 표시된다.

12.3 Type 2 우중도절단 자료(정수중단자료)

n 개의 표본을 가지고 시험을 시작하여, 미리 정해진 $r (\leq n)$ 번째 고장(재발)이 발생한 시점에서 시험을 중단하고 얻는 자료이다. 따라서 이 경우에는 r 개의 고장(재발)시간이 관측되고, 나머지 $(n-r)$ 개의 표본에 대해서는 고장(재발)시간이 r 번째 고장(재발)시간보다 크다는 사실만이 관측되고, 정확한 고장(재발)시간은 관측되지 않는다. 이러한 자료가 **Type 2 우중도절단 자료(정수중단자료)**이며, n 번째 고장(재발)시간에 시험이 중단되면 완전자료가 얻어진다. 정수중단의 경우에는, 시점 중단 시점이 연속확률변수로 간주되며, r 을 크게 잡으면 시험기간이 길어지게 된다. 일반적으로 규모가 작고 대량으로 생산되는 전자부품 등의 수명시험에는, 정해진 시간 대신에 미리 정해진 개수의 고장이 발생한 시점에서 시험을 중단하는 방법이 종종 사용되고 있다.

Type 1 우중도절단(정시중단)자료의 경우, 시험중단 시점이 미리 정해지기 때문에, 시험기간 동안 고장(재발) 자료가 전혀 관측되지 않을 위험이 있으나, Type 2 우중도절단(정수중단)자료의 경우, 미리 정해진 개수의 고장(재발)자료를 얻는 것이 보장되어 있다.

n 개의 표본을 가지고 시작된 시험에서 얻어지는 정수중단자료의 예는 다음과 같다.

$$t_{(1)}, \dots, t_{(r)}, t_{(r)}^+, \dots, t_{(n)}^+$$

여기에서 $t_{(1)} < t_{(2)} < \dots < t_{(n)}$ 은 관측된 r 개의 고장(재발) 시간을 순서대로 나열한 것이고, $(n-r)$ 개의 관측되지 않은 고장(재발) 시간은 $t_{(r)}^+$ 로 표시된다.

¹⁵예를 들면, 포아송 분포

12.4 임의 우중도절단 자료

불완전 자료 중에서 정시중단자료와 정수중단자료의 경우에는, 모든 표본에 대한 수명시험이 $t = 0$ 에서 동시에 시작되어, 일정기간이 경과된 시점, 또는 일정한 개수의 고장(재발)이 발생한 시점에서 시험이 중단된다. 따라서, 수명 시험이 중단되는 시점까지는 고장(재발)시간이 관측되고, 중단 시점이 아직도 가능되는 표본에 대해서는 모두 동일한 시험 중단시간(censoring time)이 관측된다. 그러나 경우에 따라서는, 고장(재발) 시간이 관측되기 이전에 각 표본의 사정에 따라 랜덤하게 시험이 중단되는 경우가 많이 발생한다.¹⁶

이러한 수명시험에는 시험중단시점이 각 표본에 따라서 서로 다르게 되며, 고장(재발)이 시험중단시점보다 먼저 발생하는 경우에만 고장(재발)시간이 관측되고, 그렇지 못한 표본에 대해서는 시험중단시간이 관측된다. 이러한 수명시험을 통하여 얻어진 고장(재발)자료는 **랜덤중단자료(randomly censored data)**가 되며, 이 경우에는 시험 대상이 되는 부품(환자)의 수명분포와 시험중단시간에 대한 분포를 동시에 고려함으로써, 고장(재발)시간에 대한 해석적인 분석이 가능하게 된다.

랜덤중단자료는 가장 일반적인 불완전자료로, 정시중단자료와 정수중단자료를 특별한 경우로 포함하고 있다. 랜덤중단자료의 경우에는 각 표본의 시험주단시간이 랜덤하게 발생하기 때문에, 각 고장(재발)의 발생시점에서의 생존함수가 $\frac{1}{n}$ 만큼 감소되는 것이 아니고, 시험 중단된 표본의 수에도 의존하게 된다. 따라서 **완전자료의 경험적 분포함수와 동일한 방법으로 표본의 수명분포를 추정할 수 없다.**

이 경우, 표본의 수명분포를 추정하기 위해 사용하는 방법으로는 piecewise exponential 방법 등도 있으나, 가장 보편적으로 많이 사용하는 추정 방법은 Kaplan-Meier에 의해 제안된 PL(Product Limit) 추정방법이 많이 사용된다.

12.4.1 Kaplan-Meier Estimation, Product Limit Estimation

n 개의 표본을 가지고 수명시험을 시행하는 경우 얻어지는 랜덤중단자료는 다음과 같은 형태로 얻어진다.

$$(t_1, \delta_1), (t_2, \delta_2), \dots, (t_n, \delta_n)$$

- $\delta_i = 0$ 이면 대응되는 t_i 는 표본 i 의 시험중단시점
- $\delta_i = 1$ 이면 대응되는 t_i 는 표본 i 의 관측된 고장(재발)시간

랜덤중단자료를 이용하여 생존함수에 대한 Kaplan-Meier 추정량을 구하기 위하여, 먼저 t_1, t_2, \dots, t_n 을 작은 것부터 큰 것으로 순차적으로 나열하여 순서 통계량을 $t_{(1)} < t_{(2)} < \dots < t_{(n)}$ 을 얻는다. 그러면 관측값과 대응되는 표본은 다음과 같이 나열된다.

$$(t_{(1)}, \delta_{(1)}), (t_{(2)}, \delta_{(2)}), \dots, (t_{(n)}, \delta_{(n)})$$

그러면 $t_{(i)} \leq t \leq t_{(i+1)}$, $i = 0, 1, \dots, n$ 에서의 생존함수 $S(t)$ 에 대한 Kaplan-Meier 추정 값은 다음과 같이 계산된다. 단, 여기에서 $t_{(0)} = 0$, $t_{(n+1)} = \infty$ 로 정의한다.

$$\bar{S}_n(t) = \prod_{\ell=1}^i \left(\frac{n-\ell}{n-\ell+1} \right)^{\delta_{(\ell)}}$$

Example 1. Kaplan-Meier 방법에 의한 생존함수 추정

$n = 5$ 인 랜덤중단자료가 다음과 같이 얻어졌다고 하자. 단, +가 표시된 것은 시험 중단 시간을 나타낸다.

$$100, 500, 240+, 350, 450+$$

즉, $(100, 1), (240, 0), (350, 1), (450, 0), (500, 1)$ 의 5개 관측값이며, Figure 13와 같이 표현할 수 있다.

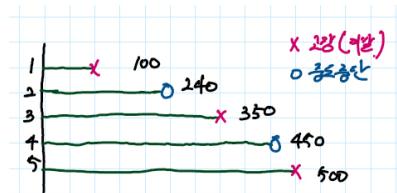


Figure 13: 랜덤 중단 자료

이 때 생존함수 $\bar{S}_n(t)$ 는 다음과 같이 구할 수 있다.

$0 \leq t < 100$ 일 때, $\bar{S}_5(t) = \frac{5}{5} = 1$ 이 된다. Kaplan-Meier 추정 값에 대한 공식을 이용하면, 구간별 생존 함수는 다음과 같이 추정된다.

$$\begin{aligned} 100 \leq t < 350 & \quad \bar{S}_5(t) = \left(\frac{4}{5} \right)^1 = 0.80 \\ 350 \leq t < 500 & \quad \bar{S}_5(t) = \left(\frac{4}{5} \right)^1 \left(\frac{3}{4} \right)^0 \left(\frac{2}{3} \right)^1 = 0.53 \\ 500 \leq t & \quad \bar{S}_5(t) = \left(\frac{4}{5} \right)^1 \left(\frac{3}{4} \right)^0 \left(\frac{2}{3} \right)^1 \left(\frac{1}{2} \right)^0 \left(\frac{0}{1} \right)^1 = 0.00 \end{aligned}$$

위 결과로부터, $\bar{R}_5(300) = 0.8$, $\bar{R}_5(400) = 0.53$ 이 된다는 사실도 알 수 있다.

¹⁶임상실험의 경우, 실험 중에 있는 환자가 다른 병원으로 옮긴다거나, 치료를 중도거부하는 등의 이유로 해서 실험에 빠지는 경우가 있다. 전자부품의 경우, 시험 중의 부품이 마모에 의한 고장이 발생하기 이전에 사용자의 과오나 외부 충격, 부품의 분실 등에 의해 시험이 중단되는 경우가 있다.

12.4.2 Life-table Method

Life-Table Method(생명표 방법)에서는, 측정 단위가 일정한 폭을 가진 구간¹⁷이며, 보험과 신뢰성 분석 자료에서 널리 사용되었다.

Example 2. Life-table Method: 심근경색 환자의 임상시험결과

Table 2는 심근경색 환자(myocardial infarction:MI)와 연관된 임상시험결과를 요약한 자료이다. 여기서 자료는 1년 단위로 생존 환자수, 사망 환자수, 그리고 중도절단된 환자수를 보여 준다.

Year(I_i)	생존 환자 수 (위험 그룹 수, n_i)	사망 환자 수 (d_i)	중도절단 환자 수 (m_i)
[0, 1)	146	27	3
[1, 2)	116	18	10
[2, 3)	88	21	10
[3, 4)	57	9	3
[4, 5)	45	1	3
[5, 6)	41	2	11
[6, 7)	28	3	5
[7, 8)	20	1	8
[8, 9)	11	2	1
[9, 10)	8	2	6

Table 2: 심근경색 환자 자료

위 표에서

- m_i : 구간 i 에서 중도절단된 환자 수
- d_i : 구간 i 에서 사망 환자 수
- n_i : 구간 i 에서 위험 환자 수, $n_i = n_1 - \sum_{j=1}^{i-1} (d_j + m_j)$

생존 함수를 구하기 위해, 다음의 조건부 확률을 고려한다.

$$q_i = P[T \in I_i | T \geq I_{i-1}] = \frac{d_i}{\binom{n_i - m_i}{2}}$$

여기서 q_i 는 $i-1$ 번째 구간까지 살아 있다가, i 번째 구간에서 사망할 확률을 의미한다.

이 때, i 번째 구간에서 m_i 명이 중도절단되었을 때, 그들 중 절반은 생존하였다고 가정한다.

따라서, i 번째 구간에서 위험에 노출된 사람 수는 $\frac{n_i - m_i}{2}$ 로, i 번째 구간에서 생존한 사람 수에서 그 구간에서 중도절단된 환자 수의 절반을 뺀 수로 정의한다.

따라서, i 번째 구간에서 생존할 확률은 $1 - q_i$ 가 된다.

i 번째 구간(I_i)에서 생존하기 위해서는, 처음부터 그 구간까지 계속 생존함을 의미하므로, 조건부 확률의 곱 공식(multiplicative formulae)에 의해 i 번째 구간에서 생존할 확률은 다음과 같다.

$$\begin{aligned}\hat{S}(I_i) &= (1 - q_1)(1 - q_2) \cdots (1 - q_i) \\ &= \prod_{j=1}^i (1 - q_j) \\ &= \prod_{j=1}^i \left[1 - \frac{d_j}{\binom{n_j - m_j}{2}} \right]\end{aligned}$$

이제 $\hat{S}(I_i)$ 의 분산을 고려해보자. $\hat{S}(I_i)$ 는 i 개의 확률($1 - q_j$, $j = 1, \dots, i$)들의 곱 형태로 표현되므로, $\hat{S}(I_i)$ 의 분산을 구하기 위해 먼저 로그 변환을 한다.

$$\log \hat{S}(I_i) = \log(1 - q_1) + \log(1 - q_2) + \cdots + \log(1 - q_i) = \hat{\theta}$$

따라서, $\{\log(1 - q_j), j = 1, \dots, i\}$ 의 분산을 이용하여, $\log S(I_i)$ 의 분산을 구할 수 있다.

여기서, $Var[\log(1 - q_j)] = \frac{d_j}{\binom{n_j - m_j}{2} - d_j} \binom{n_j - m_j}{2}$ 으로, Delta Method를 이용하여 $\hat{S} = e^{\hat{\theta}}$ 의 분산을 유도한다.

$$\begin{aligned}\widehat{Var}[\hat{S}(I_i)] &= \hat{S}^2(I_i) [Var\{\log(1 - q_1)\} + Var\{\log(1 - q_2)\} + \cdots + Var\{\log(1 - q_i)\}] \\ &= \hat{S}^2(I_i) \sum_{j=1}^i \left[\frac{d_j}{\binom{n_j - m_j}{2} - d_j} \binom{n_j - m_j}{2} \right]\end{aligned}$$

¹⁷예를 들어, 월, 분기, 년 등

13 Truncation

생존분석 시 나오는 용어로 truncation이라는 것이 있다.

이 truncation은 censoring과는 다른 기전에 의해 발생되는데, 어떤 관찰된 특정한 시간의 전/후에 사건이 발생하는 개인에게만 생긴다.

Truncation이란, 어떤 시간 간격(T_L, T_R)에서만 생존할 것으로 예측되는 개인에게서만 발생되는 sampling bias이다. 여기서 $Y_R = \text{inf}$ 이면 left truncation, $Y_L = 0$ 이면 right truncation이라고 한다.

예를 들면, 고령의 퇴직자들이 거주하는 community에 거주하는 거주자들이 사망하는 데까지 걸리는 시간을 연구하고자 할 때, 여기에 거주하는 대상자는 특정 연령이 되어야만 들어올 수 있으며, 또한 어떤 특정 연령이 되기 전에 사망한 사람들은 관찰 대상에서 제외되어야 하므로 left truncation이 발생한 것이다.

Right truncation의 예는, 1978년 4월 1일 이후에 HIV에 감염되었고, 1987년 3월 31일까지 AIDS가 발병된 258명을 대상으로 연구를 한다면, 이 기간에 HIV에 감염되었고 아직 AIDS가 발병하지 않은 대상자는 연구에서 제외되어야 한다. Event of interest가 HIV에 감염되고 AIDS가 발병하는 기간인 induction time이라고 하면, 이런 특정 기간에 감염된 대상자의 induction은 감염된 시간부터 1987년 3월 31일까지 만큼의 기간 right truncation된 것이다.

Part VI

모수적 방법을 이용한 생존함수의 추정

14 확률분포

일반적으로 통계 자료 분석에서 가장 많이 사용되는 확률분포는 정규분포로, 모든 고전적인 통계 이론과 응용 분야에서 중심적인 역할을 하고 있다. 특히, 표본의 크기가 비교적 큰 경우에는, 중심극한정리에 의하여 정규분포의 중요성이 특히 강조된다. 그러나 음수가 아닌 수명을 확률변수로 고려하는 생존분석과 수명시험에서는, 정규분포를 사용하는 것이 비현실적이다. 따라서 수명 모형 분야에서는 정규분포가 아닌 다른 분포들이 중심적인 역할을 한다.

생존함수의 추정은, 표본으로부터 수명 특성치를 구하는 것으로, 크게 점추정과 구간추정으로 나뉜다.

- 점추정: 모수의 추정값으로써 하나의 수치를 주어진 자료로부터 계산하여 구하고, 오차의 한계를 제공한다.
- 구간추정: 위 점추정에서의 오차의 한계 개념을 이용하여, 모수가 속하게 될 범위를 추정한다.

일반적으로 좋은 추정치는 다음과 같은 성질을 갖는다.

- 불편성(unbiasedness)

모수 θ 의 모든 참값에 대해 $E[\hat{\theta}] = \theta$ 이면, $\hat{\theta}$ 를 θ 의 불편추정량(unbiased estimator)이라고 한다.

- 일치성(consistency)

표본 크기와 관계되는 성질로, 표본 크기가 증가함에 따라 일치 추정량은 모수의 참값에 더욱 가까워지며,

$$\lim_{n \rightarrow \infty} P[|\hat{\theta}_n - \theta| < \epsilon] = 1$$

이 성립하면, $\hat{\theta}_n$ 은 일치성을 갖는다고 한다.

- 유효성(efficiency)

$Var[\hat{\theta}] = E[(\hat{\theta} - \theta)^2]$ 의 값이 최소가 되는 불편추정량 $\hat{\theta}$ 로, 표준오차가 적은 추정량일수록 추정의 정도가 높은 추정량이라 할 수 있다.

- 충분성(sufficiency)

$$\prod_{i=1}^n f(x_i; \theta) = f(\hat{\theta}, \theta) h(x_1, x_2, \dots, x_n)$$

즉, $f(x_i, \theta)$ 의 결합밀도함수가 $\hat{\theta}$, θ 의 함수 g 와, x_1, x_2, \dots, x_n 의 함수 h 로 인수분해될 수 있는 경우로, 표본이 갖고 있는 모수에 대한 정보 모두를 이용하는 추정치를 말한다.

참고로, 충분통계량 T 와 불편추정량 $\hat{\theta}$ 가 있을 때, 조건부 기대값 $E[\hat{\theta}|T]$ 을 취하면, 이것이 충분통계량의 함수인 불편추정량이 되고, 동시에 분산이 줄어든다. (Rao-Blackwell 정리: Note 8 참고.) 그런데, 어떤 불편추정량 $\hat{\theta}$ 으로 시작하던 $E(\hat{\theta}|T)$ 는 $\hat{\theta}$ 에 관계 없이 같은 것이 되고, 따라서 이렇게 구한 $E(\hat{\theta}|T)$ 는 분산이 최소가 된다. 즉, 최소분산 불편추정량(MVUE; Minimum Variance Unbiased Estimator)이 된다.

문제는 $E(\hat{\theta}|T)$ 를 구하는 과정이 때로는 복잡할 수 있고, 따라서 가장 순쉬운 방법은 충분통계량 T 의 함수들 가운데 불편추정량을 찾아보는 것이며, 이것마저 여의치 않다면, 번거롭더라도 $E(\hat{\theta}|T)$ 를 구할 수밖에 없다.

좀 더 엄밀한 의미에서, MVUE임을 보이려면 통계량이 최소 충분성(minimal sufficiency)을 갖고 있음을 보여야 할 필요가 있다. 이의 근거는 라오-블랙웰 정리의 도움을 받을 수 있다.

Note 8. 라오-블랙웰(Rao-Blackwell) 정리

$\hat{\theta}$ 이 θ 의 불편 추정량이고, T 가 충분통계량이라고 하자. $\theta^* = E[\hat{\theta}|T]$ 라 정의하면

- θ^* 는 θ 의 불편추정량이다.

Proof: (X_1, \dots, X_n) 이 확률표본이면, $\hat{\theta}$ 는 (X_1, \dots, X_n) 의 함수이고, T 가 충분통계량이므로 $T = t$ 가 주어졌을 때, (X_1, \dots, X_n) 의 분포는 θ 와 무관하다. 따라서 $E[\hat{\theta}|T]$ 는 T 의 함수로 θ 와 무관하다. 즉, θ^* 는 통계량이다.

그런데 다음에 따라, θ^* 는 θ 의 불편추정량이 된다.

$$E[\theta^*] = E[E(\hat{\theta}|T)] = E(\hat{\theta}) = \theta$$

- $Var(\theta^*) \leq Var(\hat{\theta})$ 이다.

Proof:

$$Var(\hat{\theta}) = Var[E(\hat{\theta}|T)] + E[Var(\hat{\theta}|T)] = Var(\theta^*) + E[Var(\hat{\theta}|T)]$$

여기서 $Var[E(\hat{\theta}|T)]$ 는 분산이기 때문에, 모든 t 에 대해서 $Var[E(\hat{\theta}|T)] \geq 0$ 이 성립하고, 따라서 $E[Var(\hat{\theta}|T)]$ 이 되므로, $Var(\theta^*) \leq Var(\hat{\theta})$ 이 성립한다.

여기에서 파라메터에 대한 notation을 다음과 같이 사용하였다.

- $a \in \mathbf{R}$: Location Parameter
- $b > 0$: Scale Parameter
- 그 외의 라틴 문자, 혹은 그리스 문자: Shape Parameter

또한, 특별한 함수에 대해서는 다음과 같이 정의하였다.

- Complete beta function

$$B(c, d) = \frac{\Gamma(c)\Gamma(d)}{\Gamma(c+d)} = \int_0^1 t^{c-1}(1-t)^{d-1} dt$$

- Incomplete beta function

$$B(P, c, d) = \frac{\Gamma(c)\Gamma(d)}{\Gamma(c+d)} = \int_0^P t^{c-1}(1-t)^{d-1} dt$$

- Complete gamma function

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt$$

- Incomplete gamma function

$$\Gamma(a, u) = \int_0^u t^{a-1} e^{-t} dt$$

- Complementary incomplete gamma function

$$\gamma(a, u) = \int_u^\infty t^{a-1} e^{-t} dt$$

- CDF of the standardized normal distribution

$$\Phi(u) = \int_{-\infty}^u \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} du$$

- PDF of the standardized normal distribution

$$\phi(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$$

- Error function

$$\text{erf}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$$

14.1 Alpha Distribution(알파 분포)

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{be^{-\frac{1}{2}(\alpha - \frac{b}{t})^2}}{\sqrt{2\pi}\Phi(\alpha)t^2}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \frac{\Phi(\alpha - \frac{b}{t})}{\Phi(\alpha)}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{be^{-\frac{1}{2}(\alpha - \frac{b}{t})^2}}{\sqrt{2\pi}[\Phi(\alpha) - \Phi(\alpha - \frac{b}{t})]t^2}$	IDHR
변수		$t \geq 0$	
파라메터		$\alpha \in \mathbf{R}, b > 0$	

Table 3: Alpha 분포함수에 기반한 척도 함수

화률변수 $Y \sim N(y; \mu, \sigma)$ 라고 하자. 다만, $y = 0$ 의 왼쪽으로 절삭되어있다.
그리면, $X = \frac{1}{Y} \sim \text{alpha}(\alpha; \alpha, b)$ 이며, 이 때 $\alpha = \frac{\mu}{\sigma}$, $b = \frac{1}{\sigma^2}$ 이다.
이 분포는 Salvia(1985)[59]가 가속 수명 모형(accelerated life testing)에 적용한 바 있다.

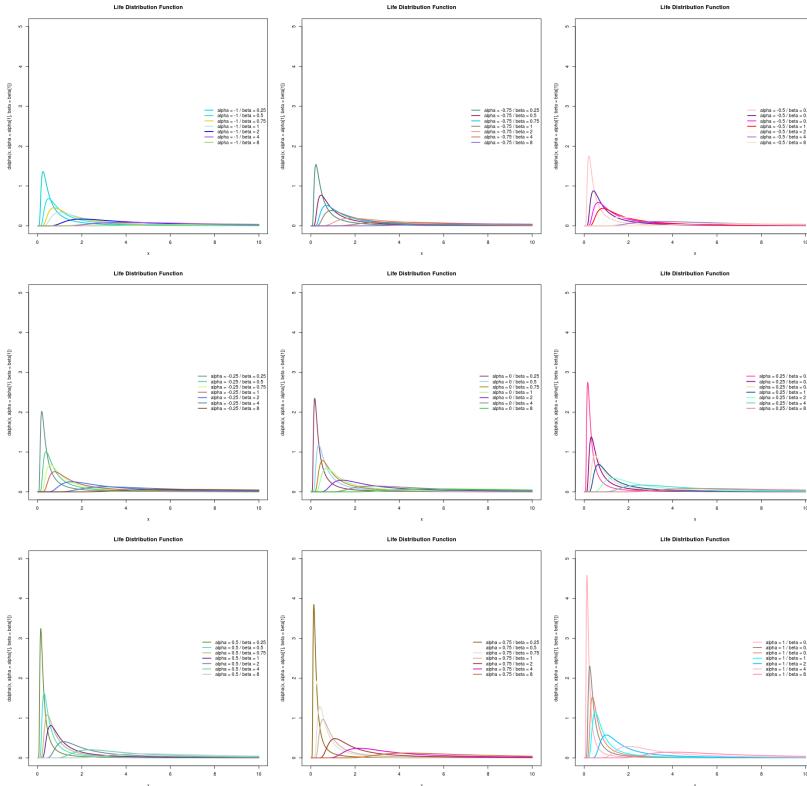


Figure 14: Alpha Distribution에 기반한 수명분포함수

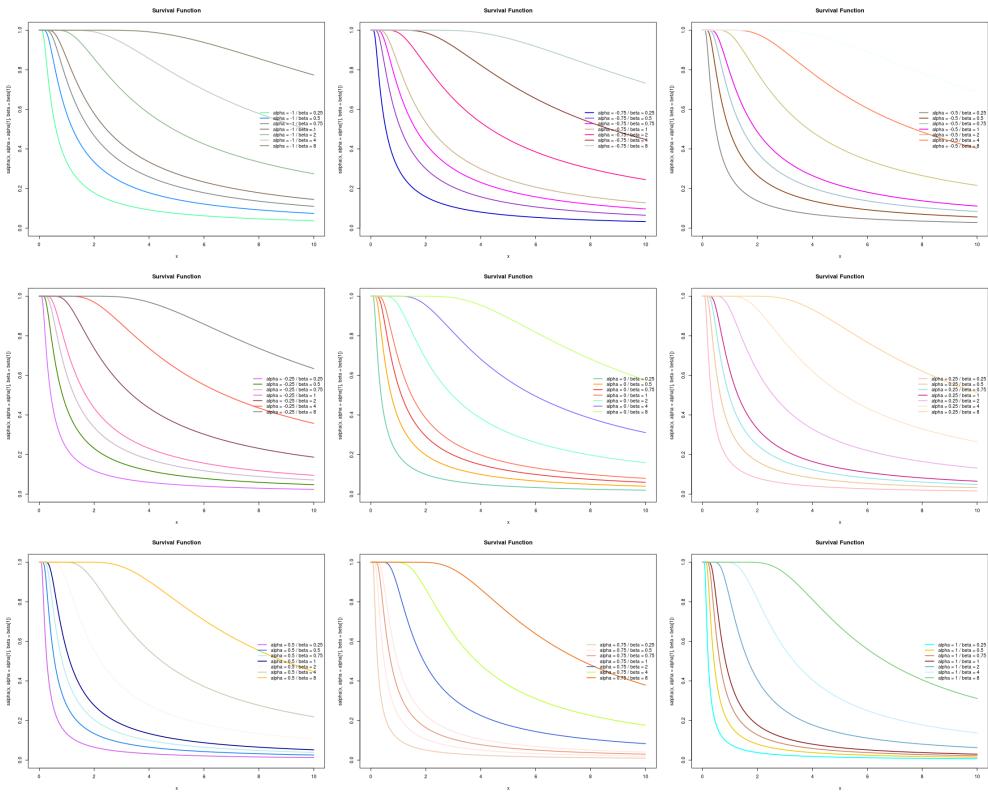


Figure 15: Alpha Distribution에 기반한 생존함수

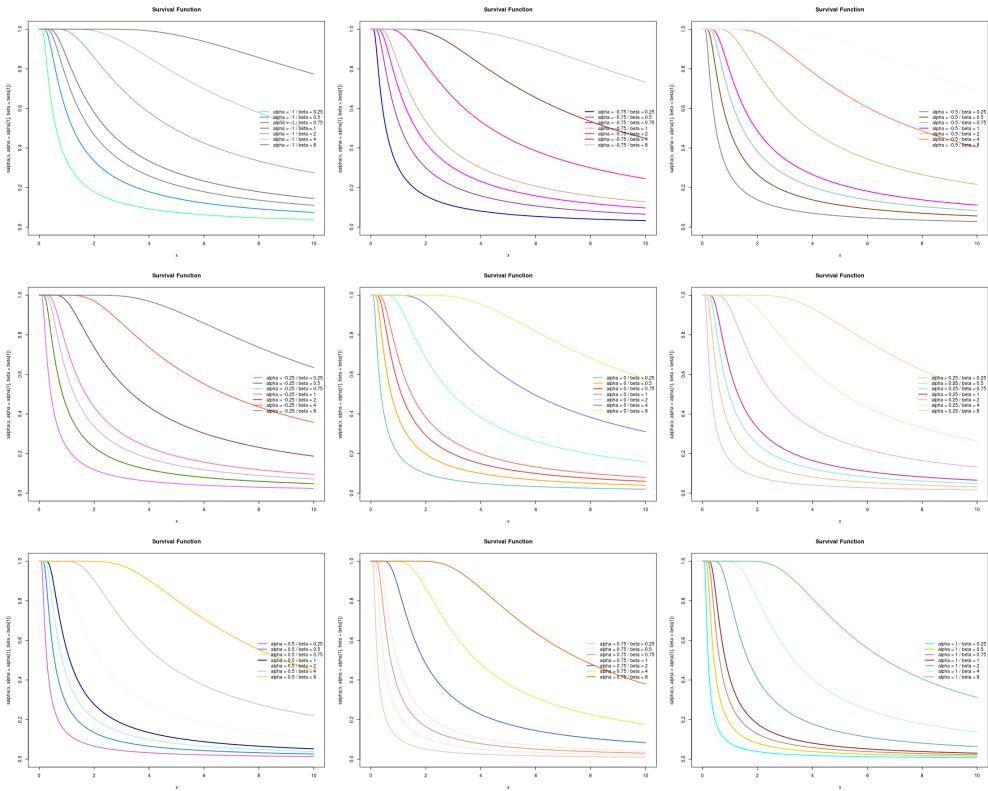


Figure 16: Alpha Distribution에 기반한 위험함수

Code 1. Alpha Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Alpha Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dalpha = function(x, alpha = 1, beta = 0)
16 {
17   if (sum((beta <= 0) * 1) > 0) { stop("beta is not positive.") } # beta > 0 이어야 한다.
18   if (sum((x < 0) * 1) > 0) { stop("x is not positive or 0.") } # x >= 0 이어야 한다.
19
20   fx = (beta * exp(-(1/2) * (alpha - beta / x)^2)) / (sqrt(2 * pi) * pnorm(alpha) * x^2)
21   return(fx)
22 }
23
24
25 ### 누적분포함수
26 palpha = function (x, alpha = 1, beta = 0)
27 {
28   if (sum((beta <= 0) * 1) > 0) { stop("beta is not positive.") } # beta > 0 이어야 한다.
29   if (sum((x < 0) * 1) > 0) { stop("x is not positive or 0.") } # x >= 0 이어야 한다.
30
31   fx = pnorm(alpha - beta/x) / pnorm(alpha)
32   return(fx)
33 }
34
35
36 ### 생존함수
37 salpha = function (x, alpha = 1, beta = 0)
38 {
39   if (sum((beta <= 0) * 1) > 0) { stop("beta is not positive.") } # beta > 0 이어야 한다.
40   if (sum((x < 0) * 1) > 0) { stop("x is not positive or 0.") } # x >= 0 이어야 한다.
41
42   fx = 1 - (pnorm(alpha - beta/x) / pnorm(alpha))
43   return(fx)
44 }
45
46
47 ### 위험함수
48 halpha = function (x, alpha = 1, beta = 0)
49 {
50   if (sum((beta <= 0) * 1) > 0) { stop("beta is not positive.") } # beta > 0 이어야 한다.
51   if (sum((x < 0) * 1) > 0) { stop("x is not positive or 0.") } # x >= 0 이어야 한다.
52
53   fx = dalpha(x, alpha, beta) / salpha(x, alpha, beta)
54   return(fx)
55 }
56
57
58
59
60
61 ##### Plot
62 plot.alpha_seq = function(x, alpha = 1, beta = 0, xlim=c(0, 10), ylim=c(0, 5), func="dalpha")
63 {
64   color=colorPalette(300)
65
66   len_alpha = length(alpha) # alpha 파라미터의 길이
67   len_beta = length(beta) # beta 파라미터의 길이
68
69   color_counter = 1
70   for (i in 1:len_alpha) ### 파라미터: alpha
71   {
72     color_counter_init = color_counter
73     legend_name = NULL;
74
75     if (func=="dalpha") # 수명분포
76     {
77       plot(x, dalpha(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[i], lwd=2, type = 'n', main="Life
78           Distribution Function")
79       for (j in 1:len_beta) ### 파라미터: beta
80       {
81         lines(x, dalpha(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
82         color_counter = color_counter + 1;
83         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
84       }
85     }
86     else if (func == "palpha") # 누적분포함수
87     {
88       plot(x, palpha(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="


```

```

88     Cumulative Distribution Function")
89     for (j in 1:len_beta) ### 파라미터: beta
90     {
91         lines(x, palpha(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
92         color_counter = color_counter + 1;
93         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
94     }
95     else if (func == "salpha") # 생존함수
96     {
97         plot(x, salpha(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival
98             Function")
99         for (j in 1:len_beta) ### 파라미터: beta
100        {
101            lines(x, salpha(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
102            color_counter = color_counter + 1;
103            legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
104        }
105     else if (func == "halpha") # 위험함수
106     {
107         plot(x, halpha(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
108             Function")
109         for (j in 1:len_beta) ### 파라미터: beta
110        {
111            lines(x, halpha(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
112            color_counter = color_counter + 1;
113            legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
114        }
115     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
116   }
117 }
118
119 par(mfrow = c(3, 3))
120 plot.alpha_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dalpha")
121
122 par(mfrow = c(3, 3))
123 plot.alpha_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="palpha")
124
125 par(mfrow = c(3, 3))
126 plot.alpha_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="salpha")
127
128 par(mfrow = c(3, 3))
129 plot.alpha_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 6), func="halpha")

```

14.2 Arcsine Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\left[b\pi \sqrt{1 - \left(\frac{t-a}{b}\right)^2} \right]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\frac{1}{2} - \frac{\arcsin\left(\frac{t-a}{b}\right)}{\pi}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\left[b\sqrt{\left(1 - \left(\frac{t-a}{b}\right)^2\right) \arccos\left(\frac{t-a}{b}\right)} \right]^{-1}$	DIHR
변수		$a - b \leq t \leq a + b$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 4: Arcsine 분포함수에 기반한 척도 함수

이 분포는 베타 분포의 두 파라메터가 $c = d = 0.5^2$ 일 때에 해당하는 특별한 경우이다. 또한 베타 분포의 두 파라메터가 $c + d = 1, c \neq 0.5$ 일 경우를 generalized arcsine distribution이라고 한다.

Arcsine distribution은 location-scale distribution의 한 종류이다.

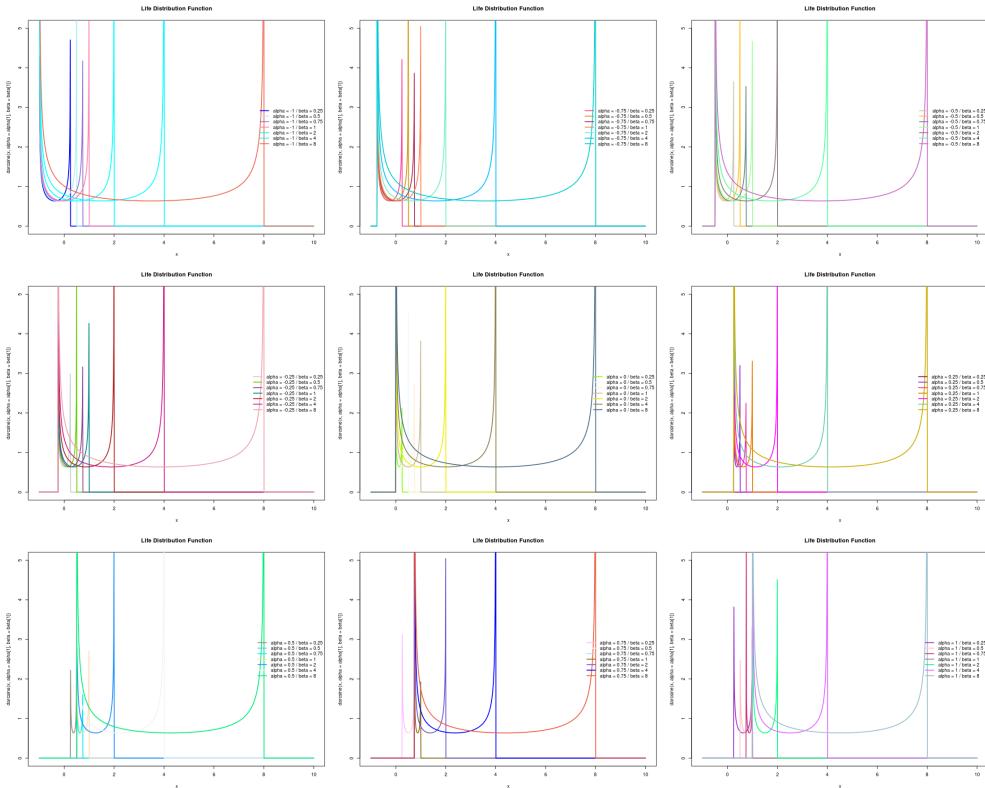


Figure 17: Arcsine Distribution에 기반한 수명분포함수

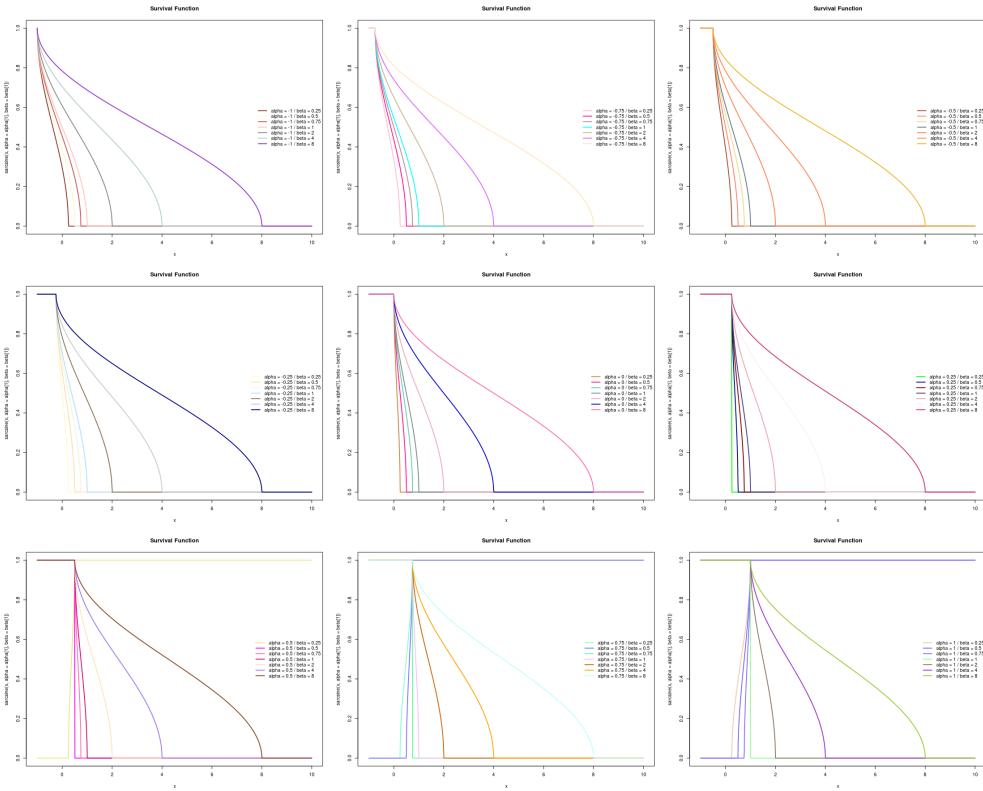


Figure 18: Arcsine Distribution에 기반한 생존함수

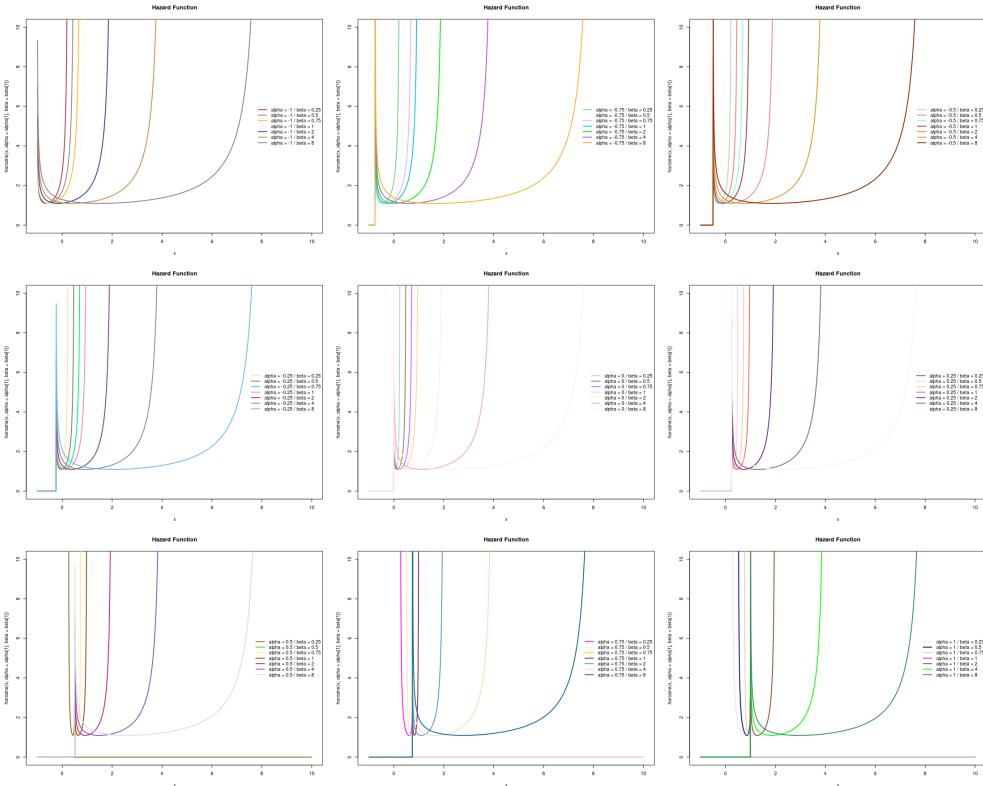


Figure 19: Arcsine Distribution에 기반한 위험함수

Code 2. Arcsine에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Arcsine Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-1, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 darcsine = function(x, alpha = 1, beta = 0)
16 {
17   fx = dbeta((x-alpha)/(beta-alpha), 0.5, 0.5)
18   return(fx)
19 }
20
21
22 ### 분위수 함수
23 qarcsine = function(x, alpha = 1, beta = 0)
24 {
25   fx = alpha + (beta-alpha) * qbeta(x, 0.5, 0.5)
26   return(fx)
27 }
28
29
30 ### 난수 함수
31 rarcsine = function(x, alpha = 1, beta = 0)
32 {
33   fx = alpha + (beta-alpha) * rbeta(x, 0.5, 0.5)
34   return(fx)
35 }
36
37
38 ### 누적분포함수
39 parcsine = function(x, alpha = 1, beta = 0)
40 {
41   fx = pbeta((x-alpha)/(beta-alpha), 0.5, 0.5)
42   return(fx)
43 }
44
45
46 ### 생존함수
47 sarcsine = function(x, alpha = 1, beta = 0)
48 {
49   fx = 1 - parcsine(x, alpha, beta)
50   return(fx)
51 }
52
53
54 ### 위험함수
55 harcsine = function(x, alpha = 1, beta = 0)
56 {
57   fx = darcsine(x, alpha, beta) / sarcsine(x, alpha, beta)
58   return(fx)
59 }
60
61
62
63
64
65 ##### Plot
66 plot.arcsine_seq = function(x, alpha = 1, beta = 0, xlim=c(0, 10), ylim=c(0, 5), func="darcsine")
67 {
68   color=colorPalette(300)
69
70   len_alpha = length(alpha) # alpha 파라미터의 길이
71   len_beta = length(beta) # beta 파라미터의 길이
72
73   color_counter = 1
74   for (i in 1:len_alpha) ### 파라미터: alpha
75   {
76     color_counter_init = color_counter
77     legend_name = NULL;
78
79     if (func=="darcsine") # 수명분포
80     {
81       plot(x, darcsine(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
82           Distribution Function")
83       for (j in 1:len_beta) ### 파라미터: beta
84       {
85         lines(x, darcsine(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
86         color_counter = color_counter + 1;
87         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
88       }
89     }
90   }
91 }
```

```

88 }
89 else if (func == "parcsine") # 누적분포함수
90 {
91   plot(x, parcsine(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
92   for (j in 1:len_beta) ### 파라메터: beta
93   {
94     lines(x, parcsine(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
95     color_counter = color_counter + 1;
96     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
97   }
98 }
99 else if (func == "sarcsine") # 생존함수
100 {
101   plot(x, sarcsine(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
102   for (j in 1:len_beta) ### 파라메터: beta
103   {
104     lines(x, sarcsine(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
105     color_counter = color_counter + 1;
106     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
107   }
108 }
109 else if (func == "harcsine") # 위험함수
110 {
111   plot(x, harcsine(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
112   for (j in 1:len_beta) ### 파라메터: beta
113   {
114     lines(x, harcsine(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
115     color_counter = color_counter + 1;
116     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
117   }
118 }
119 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
120 }
121 }
122
123 par(mfrow = c(3, 3))
124 plot.arcsine_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="darcsine")
125
126 par(mfrow = c(3, 3))
127 plot.arcsine_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="parcsine")
128
129 par(mfrow = c(3, 3))
130 plot.arcsine_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sarcsine")
131
132 par(mfrow = c(3, 3))
133 plot.arcsine_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="harcsine")

```

14.3 Beta Distribution(베타 분포)

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{B(c,d)} \cdot \frac{(t-a)^{c-1}(a+b-t)^{d-1}}{b^{c+d-1}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \frac{1}{B(c,d)} \int_0^{\frac{t-a}{b}} u^{c-1} (1-u)^{d-1} du$ $= 1 - I_{\left(\frac{t-a}{b}\right)}(c, d)$ $= I_{\left(1 - \frac{t-a}{b}\right)}(c, d)$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	DIHR for $0 < c \lesssim 0.8$ and d arbitrarily IHR for all other combinations of c and d
변수		$a \leq t \leq a + b$	
파라미터		$a \in \mathbf{R}$, $b > 0$, $c > 0$, $d > 0$	

Table 5: Beta 분포함수에 기반한 척도 함수

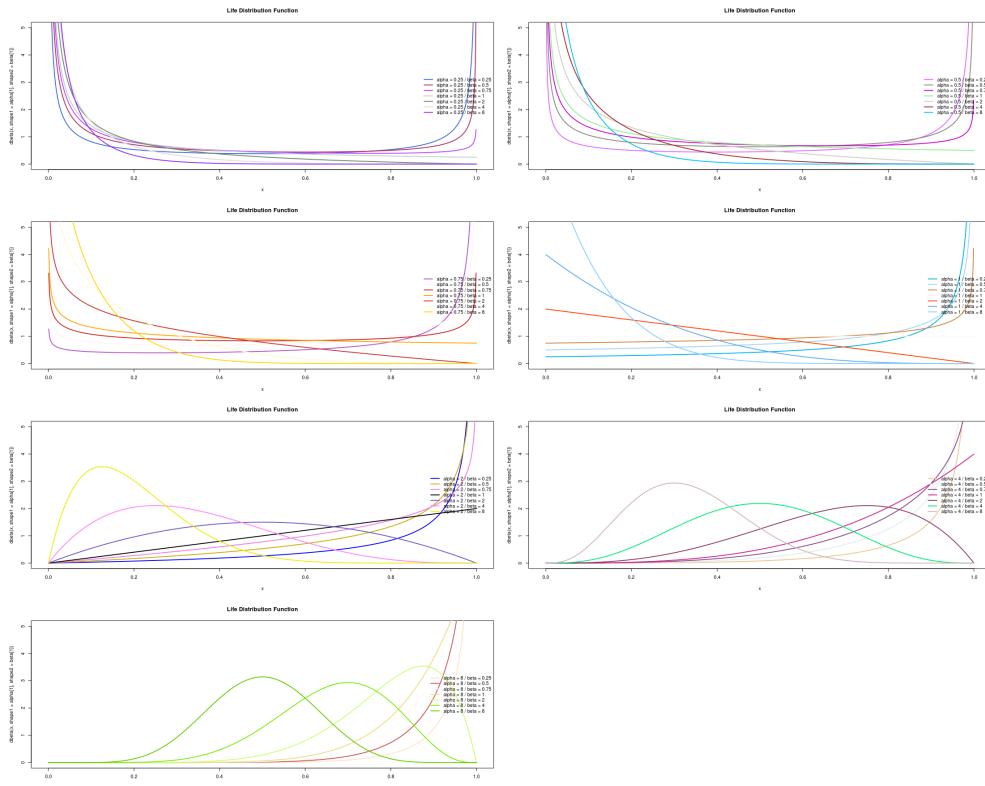


Figure 20: Beta Distribution에 기반한 수명분포함수

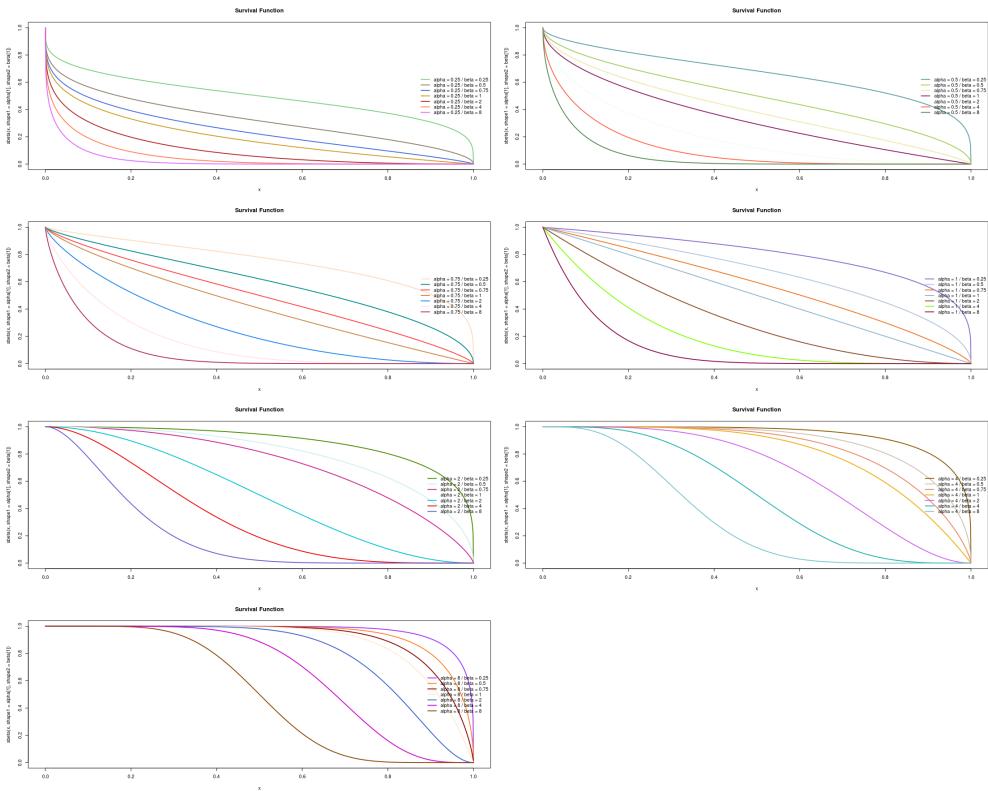


Figure 21: Beta Distribution에 기반한 생존함수

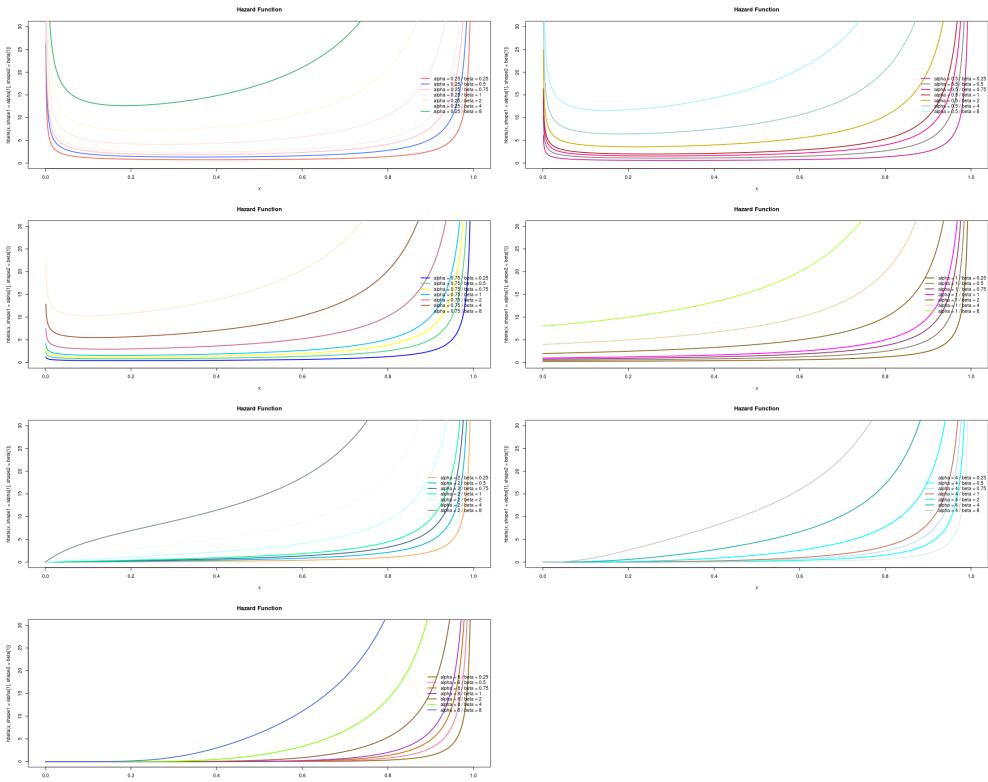


Figure 22: Beta Distribution에 기반한 위험함수

Code 3. Beta Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Beta Distribution
6 ### parameter
7 alpha = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 1, length.out = 1000)
12
13
14 ### 수명 분포
15 dbeta(x, alpha, beta)
16
17
18 ### 분위수 함수
19 qbeta(x, alpha, beta)
20
21
22 ### 난수 함수
23 rbeta(x, alpha, beta)
24
25
26 ### 누적분포함수
27 pbeta(x, alpha, beta)
28
29
30 ### 생존함수
31 sbeta = function (x, shape1 = 1, shape2 = 0)
32 {
33   fx = 1 - pbeta(x, shape1=shape1, shape2=shape2)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hbeta = function (x, shape1 = 1, shape2 = 0)
40 {
41   fx = dbeta(x, shape1=shape1, shape2=shape2) / sbeta(x, shape1=shape1, shape2=shape2)
42   return(fx)
43 }
44
45
46
47
48 ##### Plot
49 plot.beta_seq = function(x, alpha = 1, beta = 0, xlim=c(0, 10), ylim=c(0, 5), func="dbeta")
50 {
51   color=colorPalette(300)
52
53   len_alpha = length(alpha) # alpha 파라미터의 길이
54   len_beta = length(beta) # beta 파라미터의 길이
55
56   color_counter = 1
57   for (i in 1:len_alpha) ### 파라미터: alpha
58   {
59     color_counter_init = color_counter
60     legend_name = NULL;
61
62     if (func=="dbeta") # 수명분포
63     {
64       plot(x, dbeta(x, shape1=alpha[i], shape2=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
65       Distribution Function")
66       for (j in 1:len_beta) ### 파라미터: beta
67       {
68         lines(x, dbeta(x, shape1=alpha[i], shape2=beta[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
71       }
72     } else if (func == "pbeta") # 누적분포함수
73     {
74       plot(x, pbeta(x, shape1=alpha[1], shape2=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
75       for (j in 1:len_beta) ### 파라미터: beta
76       {
77         lines(x, pbeta(x, shape1=alpha[i], shape2=beta[j]), col=color[color_counter], lwd=2);
78         color_counter = color_counter + 1;
79         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
80       }
81     } else if (func == "sbeta") # 생존함수
82     {
83       plot(x, sbeta(x, shape1=alpha[1], shape2=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
84       for (j in 1:len_beta) ### 파라미터: beta
85     }
}

```

```

86      {
87        lines(x, sbeta(x, shape1=alpha[i], shape2=beta[j]), col=color[color_counter], lwd=2);
88        color_counter = color_counter + 1;
89        legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
90      }
91    } else if (func == "hbeta") # 위험함수
92    {
93      plot(x, hbeta(x, shape1=alpha[1], shape2=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
94      Function")
95      for (j in 1:len_beta) ### 파라메터: beta
96      {
97        lines(x, hbeta(x, shape1=alpha[i], shape2=beta[j]), col=color[color_counter], lwd=2);
98        color_counter = color_counter + 1;
99        legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
100     }
101   }
102   legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
103 }
104
105 par(mfrow = c(4, 2))
106 plot.beta_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dbeta")
107
108 par(mfrow = c(4, 2))
109 plot.beta_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pbeta")
110
111 par(mfrow = c(4, 2))
112 plot.beta_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sbeta")
113
114 par(mfrow = c(4, 2))
115 plot.beta_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 30), func="hbeta")
116

```

베타분포의 특징은 다음과 같다.

1. 베타 분포의 확률밀도함수는 0과 1 사이의 값에 대하여 정의된다.
2. 베타분포의 두 개의 파라메터에 따라 다음과 같은 관계를 갖게 된다.
 - $c = d = 1$: Uniform Distribution
 - $c = 2, d = 1$: The right-angled negatively skewed triangular distribution
 - $c = 1, d = 2$: The right-angled positively skewed triangular distribution
 - $c = d = 0.5$: Arcsine Distribution
 - $c > 0, d = 1$: Power Function Distribution
3. 베타분포는 유사대칭이다. 즉, 베타분포는 두 개의 모수 c 와 d 가 서로 바뀌면, 그 확률밀도함수는 원래 확률밀도함수의 mirror image가 된다.
4. 베타분포의 두 개의 파라메터에 따라 다음과 같은 모양을 하게 된다.
 - $c = d$: Symmetric
 - $c > 1, d > 1$: Unimodal
 - $c < 1, d < 1$: U-shaped
 - $d \leq 1 \leq c, c \neq d$: J-shaped
 - $c \leq 1 \leq d, c \neq d$: Inversely (or reflected) J-shaped

14.4 Birnbaum-Saunders Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{\sqrt{\frac{t}{b}} + \sqrt{\frac{b}{t}}}{2ct\sqrt{2\pi}} e^{-\frac{1}{2c^2}(\sqrt{\frac{t}{b}} - \sqrt{\frac{b}{t}})^2}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\Phi\left[-\frac{1}{c}\left(\sqrt{\frac{t}{b}} - \sqrt{\frac{b}{t}}\right)\right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	
변수		$t \geq 0$	
파라메터		$b > 0, c > 0$	

Table 6: Birnbaum-Saunders 분포함수에 기반한 척도 함수

이 분포는 Birnbaum and Saunders(1968, 1969)[16][17]에 의해 제안되었다.

$Y = \frac{\sqrt{\frac{t}{b}} - \sqrt{\frac{b}{t}}}{c} \sim N(0, 1)$ 이 된다. 생존함수 $S(t)$ 는 이에 차운하여 작성된 것이다.

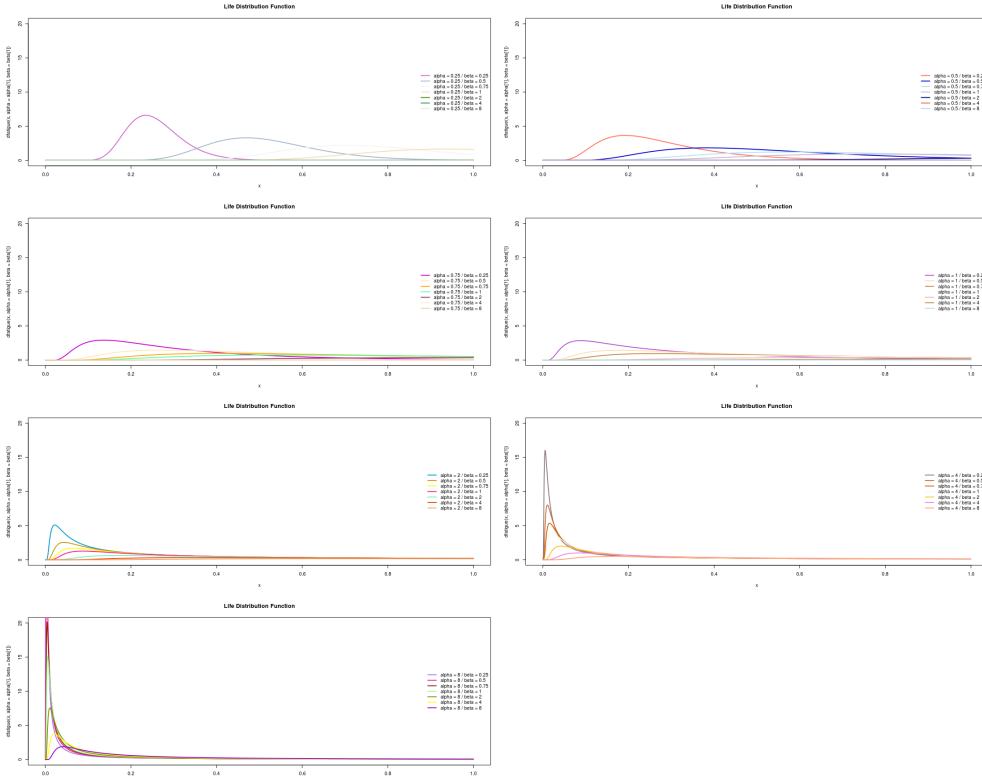


Figure 23: Birnbaum-Saunders Distribution에 기반한 수명분포함수

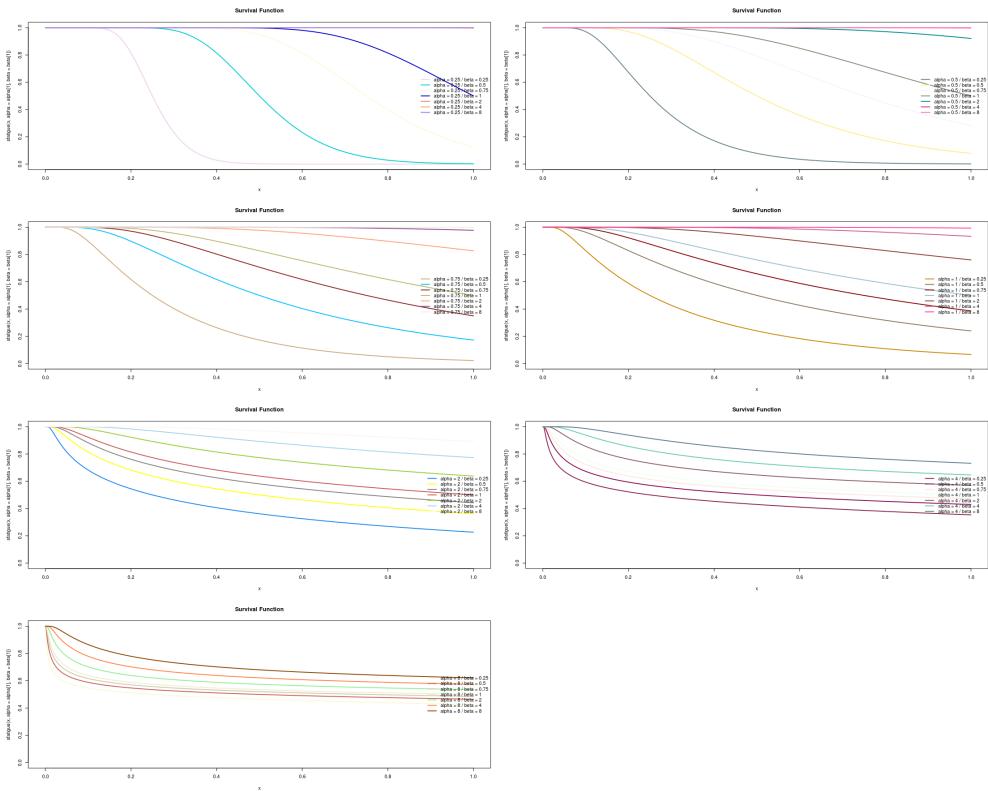


Figure 24: Birnbaum-Saunders Distribution에 기반한 생존함수

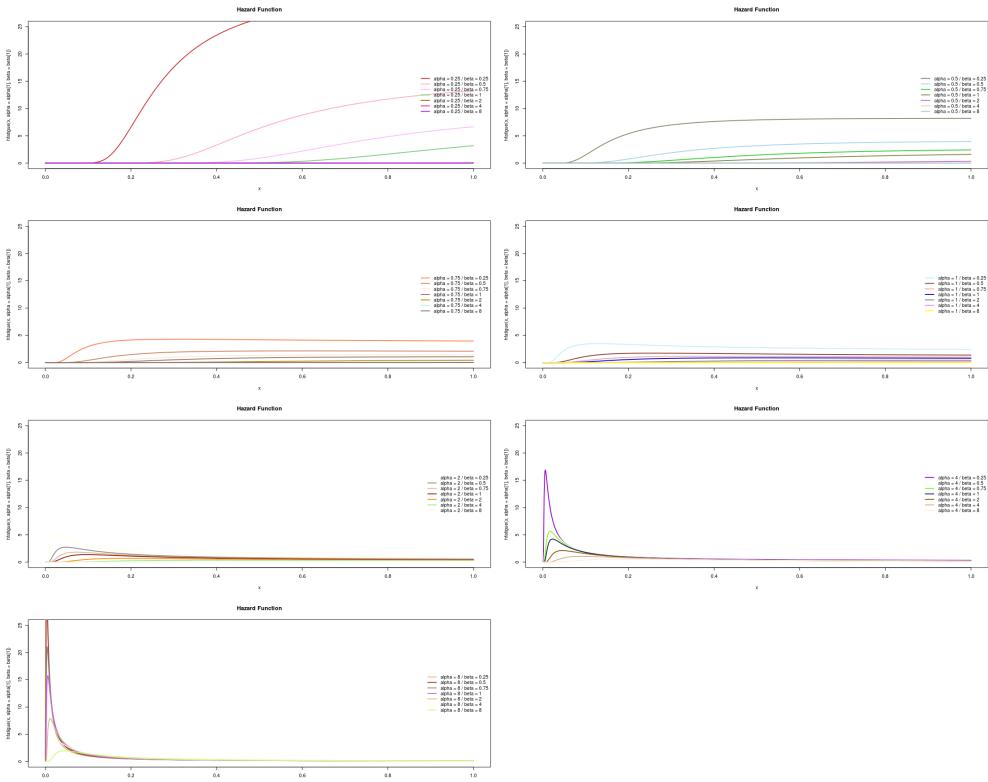


Figure 25: Birnbaum-Saunders Distribution에 기반한 위험함수

Code 4. Birnbaum-Saunders Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(extraDistr)
4
5
6 ##### Birnbaum-Saunders Distribution
7 ### parameter
8 alpha = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 1, length.out = 1000)
13
14
15 ### 수명 분포
16 dfatigue(x, alpha, beta = 1, mu = 0)
17
18
19 ### 분위수 함수
20 qfatigue(x, alpha, beta, mu = 0)
21
22
23 ### 난수 함수
24 rfatigue(x, alpha, beta, mu = 0)
25
26
27 ### 누적분포함수
28 pfatigue(x, alpha, beta, mu = 0)
29
30
31 ### 생존함수
32 sfatigue = function (x, alpha = 1, beta = 0)
33 {
34   fx = 1 - pfatigue(x, alpha, beta)
35   return(fx)
36 }
37
38
39 ### 위험함수
40 hfatigue = function (x, alpha = 1, beta = 0)
41 {
42   fx = dfatigue(x, alpha, beta) / sfatigue(x, alpha, beta)
43   return(fx)
44 }
45
46
47
48
49
50 ##### Plot
51 plot.fatigue_seq = function(x, alpha = 1, beta = 0, xlim=c(0, 10), ylim=c(0, 5), func="dfatigue")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라메터의 길이
56   len_beta = length(beta) # beta 파라메터의 길이
57
58   color_counter = 1
59   for (i in 1:len_alpha) ### 파라메터: alpha
60   {
61     color_counter_init = color_counter
62     legend_name = NULL;
63
64     if (func=="dfatigue") # 수명분포
65     {
66       plot(x, dfatigue(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
67           Distribution Function")
68       for (j in 1:len_beta) ### 파라메터: beta
69       {
70         lines(x, dfatigue(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
71         color_counter = color_counter + 1;
72         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
73       }
74     }
75     else if (func == "pfatigue") # 누적분포함수
76     {
77       plot(x, pfatigue(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
78       for (j in 1:len_beta) ### 파라메터: beta
79       {
80         lines(x, pfatigue(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
83       }
84     }
85     else if (func == "sfatigue") # 생존함수

```

```

85     {
86         plot(x, sfatigue(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
87         for (j in 1:len_beta) ### 파라미터: beta
88         {
89             lines(x, sfatigue(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
90             color_counter = color_counter + 1;
91             legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
92         }
93     }
94     else if (func == "hfatigue") # 위험함수
95     {
96         plot(x, hfatigue(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
97         for (j in 1:len_beta) ### 파라미터: beta
98         {
99             lines(x, hfatigue(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100            color_counter = color_counter + 1;
101            legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102        }
103    }
104    legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
105 }
106 }
107
108 par(mfrow = c(4, 2))
109 plot.fatigue_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 20), func="dfatigue")
110
111 par(mfrow = c(4, 2))
112 plot.fatigue_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pfatigue")
113
114 par(mfrow = c(4, 2))
115 plot.fatigue_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sfatigue")
116
117 par(mfrow = c(4, 2))
118 plot.fatigue_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 25), func="hfatigue")

```

14.5 Burr Distribution of Type XII

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{cd}{b} \left(\frac{t-a}{b} \right)^{d-1} \left[1 + \left(\frac{t-a}{b} \right)^d \right]^{-(c+1)}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\left[1 + \left(\frac{t-a}{b} \right)^d \right]^{-c}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{cd}{b} \left(\frac{t-a}{b} \right)^{d-1} \left[1 + \left(\frac{t-a}{b} \right)^d \right]^{-1}$	DHR for $0 < d \leq 1$ IDHR for $d > 1$
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0, d > 0$	

Table 7: Burr 분포함수에 기반한 척도 함수

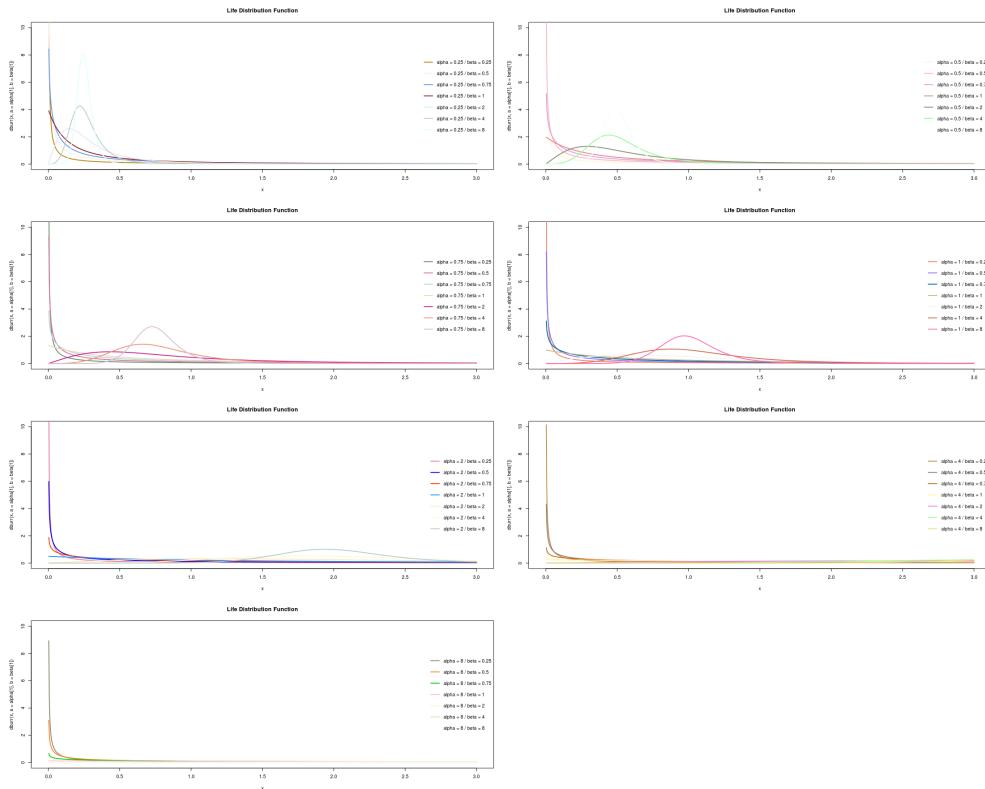


Figure 26: Burr Distribution에 기반한 수명분포함수

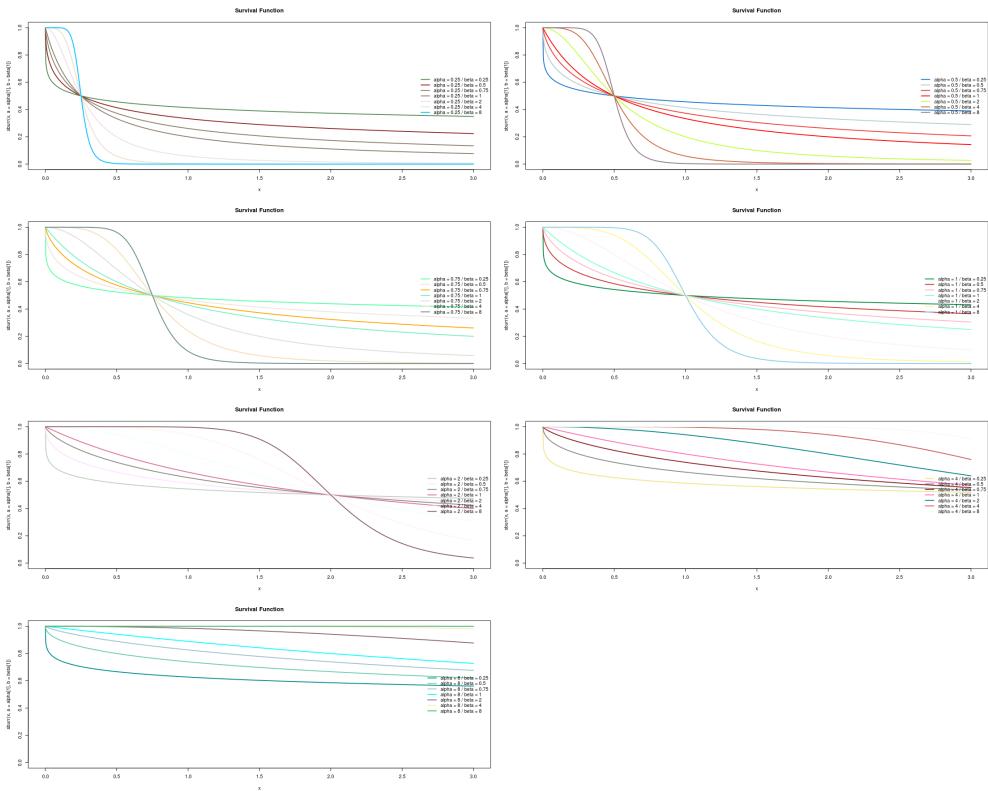


Figure 27: Burr Distribution에 기반한 생존함수

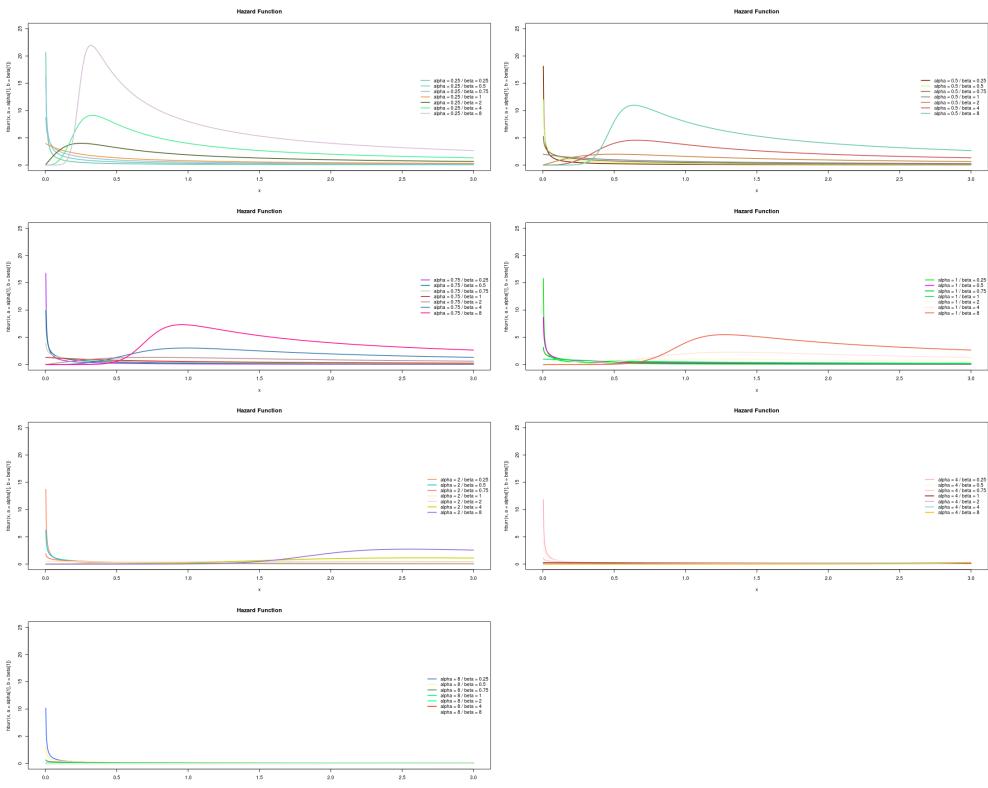


Figure 28: Burr Distribution에 기반한 위험함수

Code 5. Burr Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(extremefit)
4
5 ###### Burr Distribution
6 ### parameter
7 alpha = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input variable
11 x = seq(0, 3, length.out = 1000)
12
13
14 ### 수명 분포
15 dburr(x, a=alpha, b=beta)
16
17
18 ### 분위수 함수
19 qburr(x, a=alpha, b=beta)
20
21
22 ### 난수 함수
23 rburr(x, a=alpha, b=beta)
24
25
26 ### 누적분포함수
27 pburr(x, a=alpha, b=beta)
28
29
30 ### 생존함수
31 sburr = function (x, a = 1, b = 0)
32 {
33   fx = 1 - pburr(x, a=a, b=b)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hburr = function (x, a=a, b=b)
40 {
41   fx = dburr(x, a=a, b=b) / sburr(x, a=a, b=b)
42   return(fx)
43 }
44
45
46
47
48
49
50 ##### Plot
51 plot.burr_seq = function(x, alpha = 1, beta = 0, xlim=c(0, 10), ylim=c(0, 5), func="dburr")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라미터의 길이
56   len_beta = length(beta) # beta 파라미터의 길이
57
58   color_counter = 1
59   for (i in 1:len_alpha) ### 파라미터: alpha
60   {
61     color_counter_init = color_counter
62     legend_name = NULL;
63
64     if (func=="dburr") # 수명분포
65     {
66       plot(x, dburr(x, a=alpha[i], b=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution
67           Function")
68       for (j in 1:len_beta) ### 파라미터: beta
69       {
70         lines(x, dburr(x, a=alpha[i], b=beta[j]), col=color[color_counter], lwd=2);
71         color_counter = color_counter + 1;
72         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
73       }
74     } else if (func == "pburr") # 누적분포함수
75     {
76       plot(x, pburr(x, a=alpha[1], b=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
77           Distribution Function")
78       for (j in 1:len_beta) ### 파라미터: beta
79       {
80         lines(x, pburr(x, a=alpha[i], b=beta[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
83       }
84     } else if (func == "sburr") # 생존함수
85     {
86       plot(x, sburr(x, a=alpha[1], b=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function"

```

```

        ")
87    for (j in 1:len_beta) ### 파라미터: beta
88    {
89      lines(x, sburr(x, a=alpha[i], b=beta[j]), col=color[color_counter], lwd=2);
90      color_counter = color_counter + 1;
91      legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
92    }
93  }
94  else if (func == "hburr") # 위험함수
95  {
96    plot(x, hburr(x, a=alpha[1], b=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
97    for (j in 1:len_beta) ### 파라미터: beta
98    {
99      lines(x, hburr(x, a=alpha[i], b=beta[j]), col=color[color_counter], lwd=2);
100     color_counter = color_counter + 1;
101     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102   }
103 }
104 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
105 }
106 }
107
108 par(mfrow = c(4, 2))
109 plot.burr_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="dburr")
110
111 par(mfrow = c(4, 2))
112 plot.burr_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pburr")
113
114 par(mfrow = c(4, 2))
115 plot.burr_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sburr")
116
117 par(mfrow = c(4, 2))
118 plot.burr_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 25), func="hburr")

```

14.6 Cauchy Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\left[\pi b \left(1 + \left(\frac{t-a}{b} \right)^2 \right) \right]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\frac{1}{2} - \frac{1}{\pi} \arctan \left(\frac{t-a}{b} \right)$	
위험률 함수 (고장률 함수) 변수	$h(t) = \frac{f(t)}{S(t)}$	$\left[b \left(1 + \left(\frac{t-a}{b} \right)^2 \right) \left(\frac{\pi}{2} - \arctan \left(\frac{t-a}{b} \right) \right) \right]^{-1}$ $t \in \mathbf{R}$	IDHR
파라메터		$a \in \mathbf{R}, b > 0$	

Table 8: Cauchy 분포함수에 기반한 척도 함수

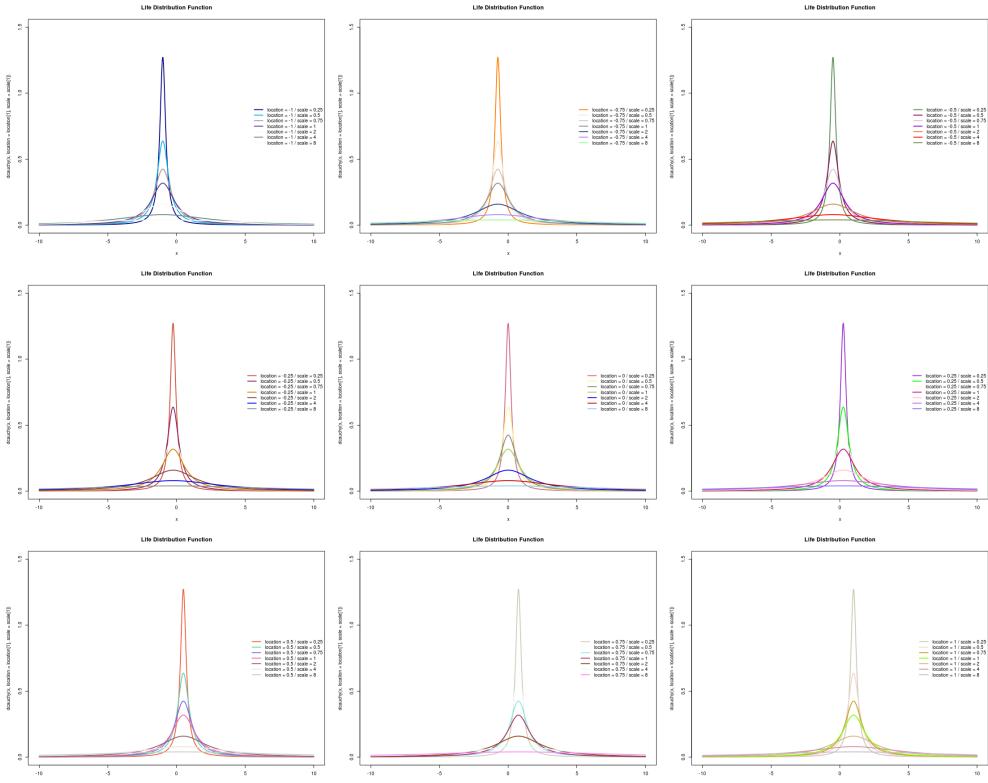


Figure 29: Cauchy Distribution에 기반한 수명분포함수

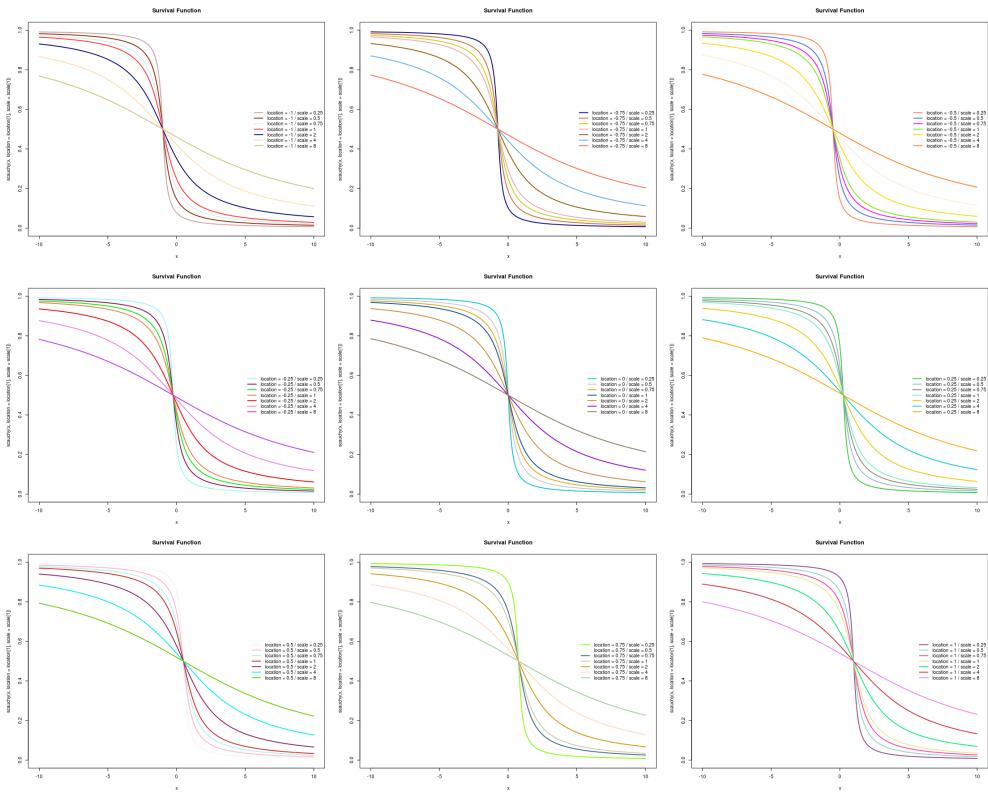


Figure 30: Cauchy Distribution에 기반한 생존함수

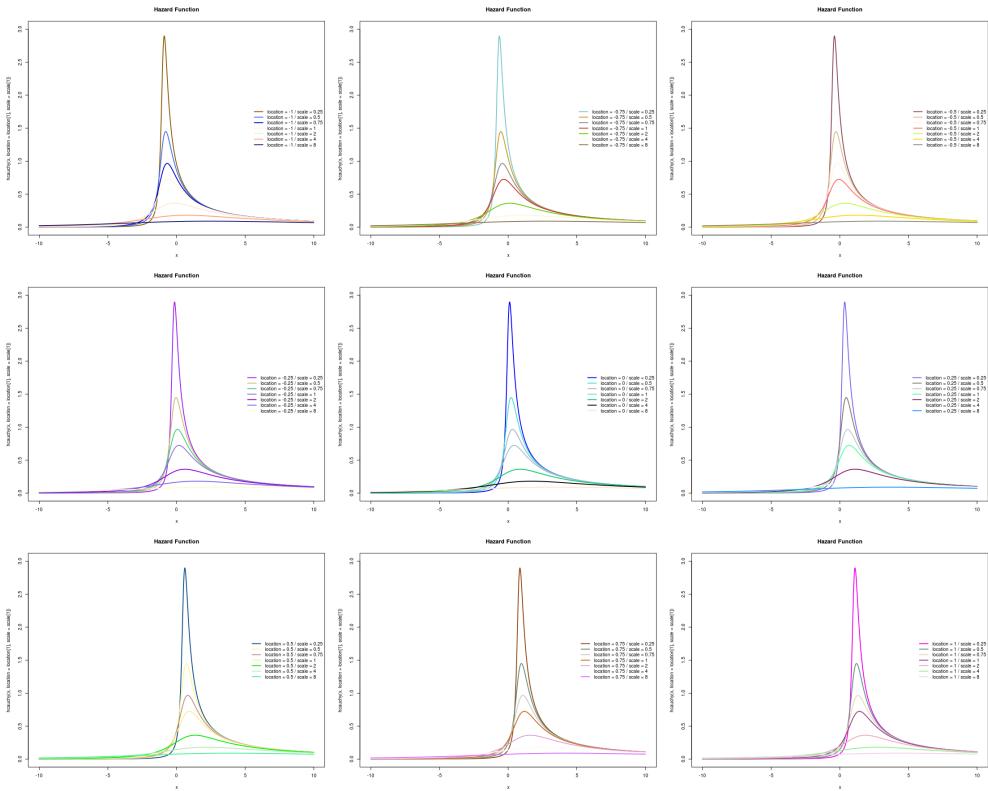


Figure 31: Cauchy Distribution에 기반한 위험함수

Code 6. Cauchy Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Cauchy Distribution
6 ### parameter
7 location = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dcauchy(x, location, scale)
16
17
18 ### 분위수 함수
19 qcauchy(x, location, scale)
20
21
22 ### 난수 함수
23 rcauchy(x, location, scale)
24
25
26 ### 누적분포함수
27 pcauchy(x, location, scale)
28
29
30 ### 생존함수
31 scauchy = function (x, location = 1, scale = 0)
32 {
33   fx = 1 - pcauchy(x, location, scale)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hcauchy = function (x, location = 1, scale = 0)
40 {
41   fx = dcauchy(x, location, scale) / scauchy(x, location, scale)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.cauchy_seq = function(x, location = 1, scale = 0, xlim=c(0, 10), ylim=c(0, 5), func="dcauchy")
51 {
52   color=colorPalette(300)
53
54   len_location = length(location) # location 파라메터의 길이
55   len_scale = length(scale) # scale 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_location) ### 파라메터: location
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dcauchy") # 수명분포
64     {
65       plot(x, dcauchy(x, location=location[i], scale=scale[i]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
66       for (j in 1:len_scale) ### 파라메터: scale
67       {
68         lines(x, dcauchy(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
71       }
72     }
73     else if (func == "pcauchy") # 누적분포함수
74     {
75       plot(x, pcauchy(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
76       for (j in 1:len_scale) ### 파라메터: scale
77       {
78         lines(x, pcauchy(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
79         color_counter = color_counter + 1;
80         legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
81       }
82     }
83     else if (func == "scauchy") # 생존함수
84     {
85       plot(x, scauchy(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
86     }
87   }
88 }
```

```

86     for (j in 1:len_scale) ### 파라미터: scale
87     {
88       lines(x, scauchy(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
91     }
92   }
93 else if (func == "hcauchy") # 위험함수
94 {
95   plot(x, hcauchy(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
96   for (j in 1:len_scale) ### 파라미터: scale
97   {
98     lines(x, hcauchy(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
99     color_counter = color_counter + 1;
100    legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
101  }
102 }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.cauchy_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1.5), func="dcauchy")
109
110 par(mfrow = c(3, 3))
111 plot.cauchy_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pcauchy")
112
113 par(mfrow = c(3, 3))
114 plot.cauchy_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="scauchy")
115
116 par(mfrow = c(3, 3))
117 plot.cauchy_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 3), func="hcauchy")

```

14.6.1 Half-Cauchy Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$2 \left[b\pi \left(1 + \left(\frac{t-a}{b} \right)^2 \right) \right]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - \frac{2}{\pi} \arctan \left(\frac{t-a}{b} \right)$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$2 \left[b \left(1 + \left(\frac{t-a}{b} \right)^2 \right) (\pi - 2 \arctan \left(\frac{t-a}{b} \right)) \right]^{-1}$	IDHR
변수		$t \geq a$	
파라미터		$a \in \mathbf{R}, b > 0$	

Table 9: Half-Cauchy 분포함수에 기반한 척도 함수

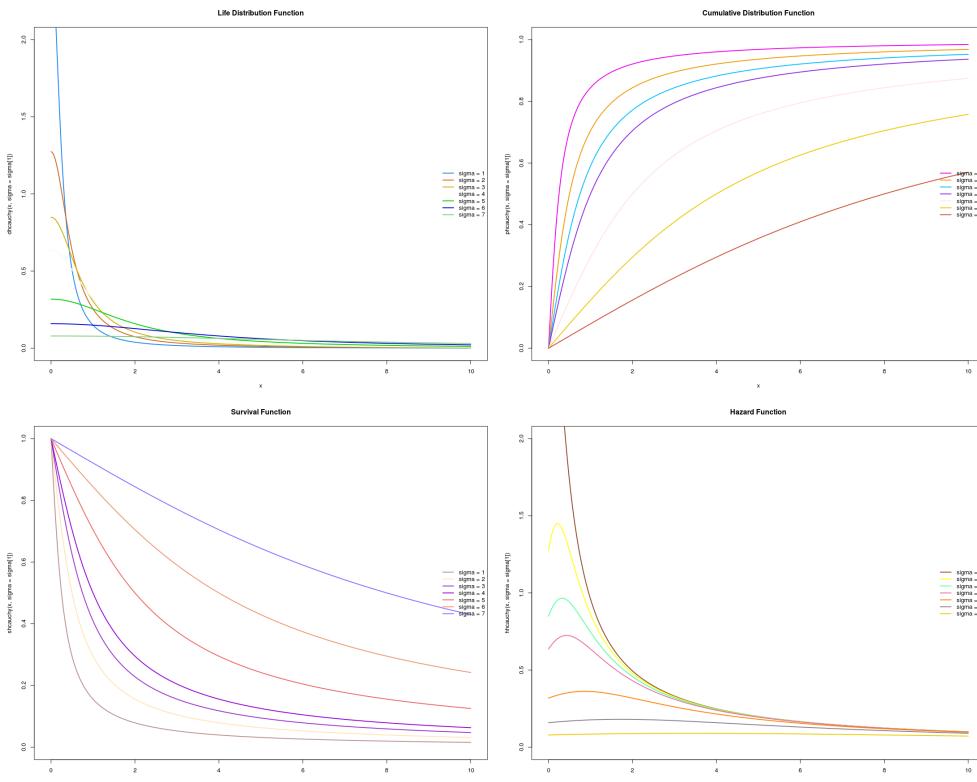


Figure 32: Half-Cauchy Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 7. Half-Cauchy Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(extraDistr)
4
5 ###### half-Cauchy Distribution
6 ### parameter
7 sigma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8
9 #### input varialbe
10 x = seq(0, 10, length.out = 1000)
11
12
13 #### 수명 분포
14 dhcauchy(x, sigma = sigma)
15
16
17 #### 분위수 함수
18 qhcauchy(x, sigma = sigma)
19
20
21 #### 난수 함수
22 rhcauchy(x, sigma = sigma)
23
24
25 #### 누적분포함수
26 phcauchy(x, sigma = sigma)
27
28
29 #### 생존함수
30 shcauchy = function (x, sigma = 1)
31 {
32   fx = 1 - phcauchy(x, sigma = sigma)
33   return(fx)
34 }
35
36
37 #### 위험함수
38 hhcauchy = function (x, sigma = 1)
39 {
40   fx = dhcauchy(x, sigma = sigma) / shcauchy(x, sigma = sigma)
41   return(fx)
42 }
43
44
45
46
47
48 ###### Plot
49 plot.hcauchy_seq = function(x, sigma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dhcauchy")
50 {
51   color=colorPalette(300)
52
53   len_sigma = length(sigma) # sigma 파라메터의 길이
54
55   color_counter = 1
56   color_counter_init = color_counter
57   legend_name = NULL;
58
59
60   if (func=="dhcauchy") # 수명분포
61   {
62     plot(x, dhcauchy(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution
Function")
63     for (i in 1:len_sigma) ### 파라메터: sigma
64     {
65       lines(x, dhcauchy(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
66       color_counter = color_counter + 1;
67       legend_name = c(legend_name, paste("sigma = ", i, sep=""))
68     }
69   }
70   else if (func == "phcauchy") # 누적분포함수
71   {
72     plot(x, phcauchy(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution
Function")
73     for (i in 1:len_sigma) ### 파라메터: sigma
74     {
75       lines(x, phcauchy(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
76       color_counter = color_counter + 1;
77       legend_name = c(legend_name, paste("sigma = ", i, sep=""))
78     }
79   }
80   else if (func == "shcauchy") # 생존함수
81   {
82     plot(x, shcauchy(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
83     for (i in 1:len_sigma) ### 파라메터: sigma
```

```

85     {
86         lines(x, shcauchy(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
87         color_counter = color_counter + 1;
88         legend_name = c(legend_name, paste("sigma = ", i, sep=""))
89     }
90 }
91 else if (func == "hhcauchy") # 위험함수
92 {
93     plot(x, hhcauchy(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
94     for (i in 1:len_sigma) ### 퍼라메터: sigma
95     {
96         lines(x, hhcauchy(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
97         color_counter = color_counter + 1;
98         legend_name = c(legend_name, paste("sigma = ", i, sep=""))
99     }
100 }
101 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
102 }
103
104 par(mfrow = c(2, 2))
105 plot.hcauchy_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dhcauchy")
106 plot.hcauchy_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 1), func="phcauchy")
107 plot.hcauchy_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 1), func="shcauchy")
108 plot.hcauchy_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 2), func="hhcauchy")

```

14.7 Chi(χ) Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$t^{\nu-1} e^{-\frac{t^2}{2}} [2^{\frac{\nu}{2}-1} \Gamma(\frac{\nu}{2})]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - \frac{\Gamma(\frac{\nu}{2}, \frac{t^2}{2})}{\Gamma(\frac{\nu}{2})}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{t^{\nu-1} e^{-\frac{t^2}{2}}}{2^{\frac{\nu}{2}-1} [\Gamma(\frac{\nu}{2}) - \Gamma(\frac{\nu}{2}, \frac{t^2}{2})]}$	DIHR for $0 < \nu < 1$ IHR for $\nu \geq 1$
변수		$t \geq 0$	
파라미터		$\nu > 0$	

Table 10: χ 분포함수에 기반한 척도 함수

파라미터 $\nu = 2$ 이면, χ 분포가 Rayleigh 분포와 같아진다.

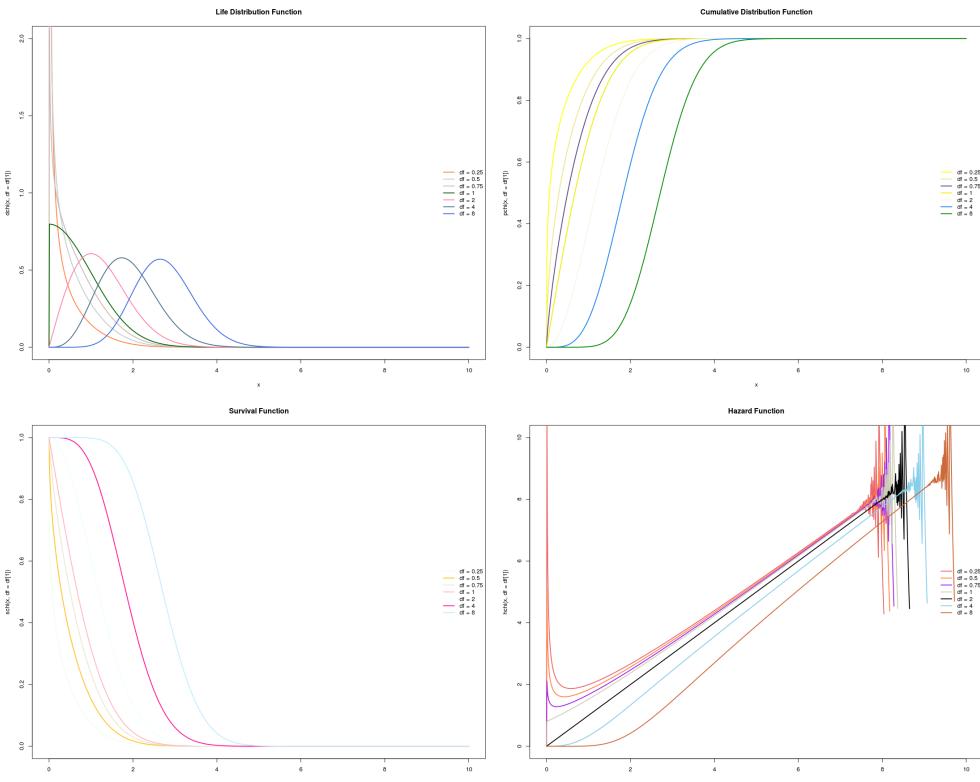


Figure 33: Chi Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 8. Chi Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(EnvStats)
4
5
6 ##### Chi Distribution
7 ### parameter
8 df = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dchi(x, df = df)
16
17
18 ### 분위수 함수
19 qchi(x, df = df)
20
21
22 ### 난수 함수
23 rchi(x, df = df)
24
25
26 ### 누적분포함수
27 pchi(x, df = df)
28
29
30 ### 생존함수
31 schi = function (x, df = 1)
32 {
33   fx = 1 - pchi(x, df = df)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hchi = function (x, df = 1)
40 {
41   fx = dchi(x, df = df) / schi(x, df = df)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.chi_seq = function(x, df = 1, xlim=c(0, 10), ylim=c(0, 5), func="dchi")
51 {
52   color=colorPalette(300)
53
54   len_df = length(df) # df 파라메터의 길이
55
56   color_counter = 1
57   color_counter_init = color_counter
58   legend_name = NULL;
59
60
61   if (func=="dchi") # 수명분포
62   {
63     plot(x, dchi(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
64     for (i in 1:len_df) ### 파라메트: df
65     {
66       lines(x, dchi(x, df=df[i]), col=color[color_counter], lwd=2);
67       color_counter = color_counter + 1;
68       legend_name = c(legend_name, paste("df = ", df[i], sep=""))
69     }
70   }
71   else if (func == "pchi") # 누적분포함수
72   {
73     plot(x, pchi(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
74     for (i in 1:len_df) ### 파라메트: df
75     {
76       lines(x, pchi(x, df=df[i]), col=color[color_counter], lwd=2);
77       color_counter = color_counter + 1;
78       legend_name = c(legend_name, paste("df = ", df[i], sep=""))
79     }
80   }
81   else if (func == "schi") # 생존함수
82   {
83     plot(x, schi(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
84     for (i in 1:len_df) ### 파라메트: df
85     {
86       lines(x, schi(x, df=df[i]), col=color[color_counter], lwd=2);
```

```

87         color_counter = color_counter + 1;
88         legend_name = c(legend_name, paste("df = ", df[i], sep=""))
89     }
90 } else if (func == "hchi") # 위험함수
91 {
92     plot(x, hchi(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
93     for (i in 1:len_df) ### 파라메트릭 df
94     {
95         lines(x, hchi(x, df=df[i]), col=color[color_counter], lwd=2);
96         color_counter = color_counter + 1;
97         legend_name = c(legend_name, paste("df = ", df[i], sep=""))
98     }
99 }
100 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
101 }
102 }
103
104 par(mfrow = c(2, 2))
105 plot.chi_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dchi")
106 plot.chi_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pchi")
107 plot.chi_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 1), func="schis")
108 plot.chi_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hchi")

```

14.8 Chi-square(χ^2) Distribution(카이제곱 분포)

Table 11: 카이제곱분포함수에 기반한 척도 함수

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$t^{\left(\frac{\nu}{2}-1\right)} e^{-\frac{t}{2}} \left[2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)\right]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - \frac{\Gamma\left(\frac{\nu}{2}, \frac{t}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{t^{\frac{\nu}{2}-1} e^{-\frac{t}{2}}}{2^{\frac{\nu}{2}} [\Gamma\left(\frac{\nu}{2}\right) - \Gamma\left(\frac{\nu}{2}, \frac{t}{2}\right)]}$	DHR for $0 < \nu < 2$ 0.5 for $\nu = 2$ IHR for $\nu > 2$
변수		$t \geq 0$	
파라메터		$\nu > 0$	

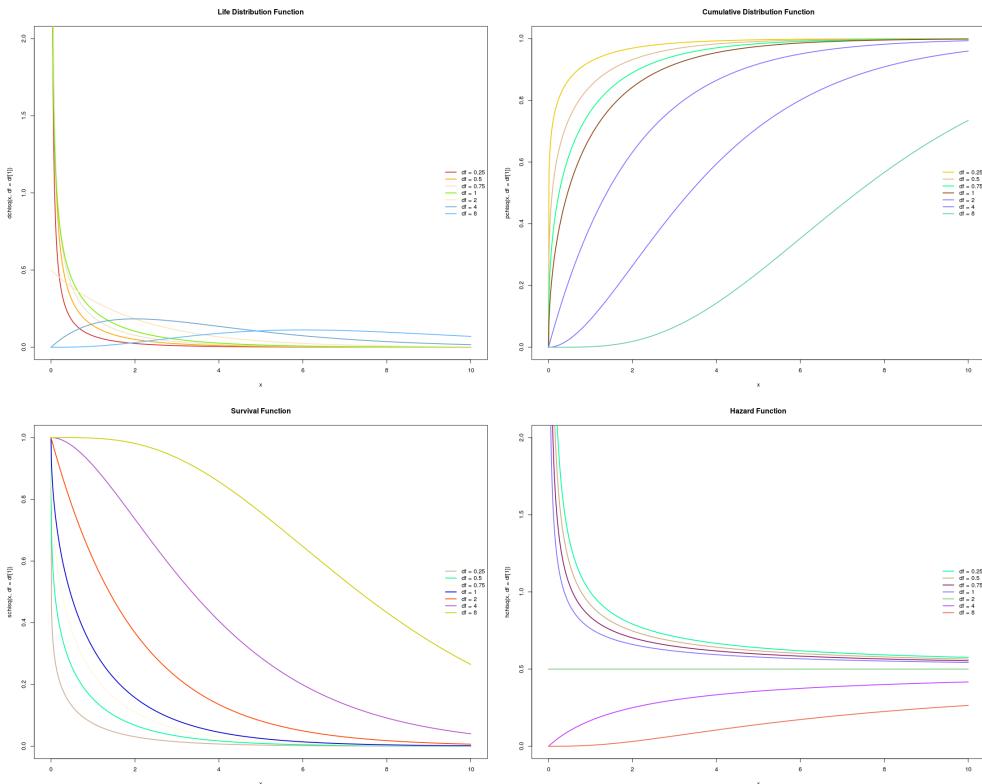


Figure 34: Chi-square Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 9. Chi-square Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Chi-square Distribution
6 ### parameter
7 df = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8
9 ### input varialbe
10 x = seq(0, 10, length.out = 1000)
11
12
13 ### 수명 분포
14 dchisq(x, df = df)
15
16
17 ### 분위수 함수
18 qchisq(x, df = df)
19
20
21 ### 난수 함수
22 rchisq(x, df = df)
23
24
25 ### 누적분포함수
26 pchisq(x, df = df)
27
28
29 ### 생존함수
30 schisq = function (x, df = 1)
31 {
32   fx = 1 - pchisq(x, df = df)
33   return(fx)
34 }
35
36
37 ### 위험함수
38 hchisq = function (x, df = 1)
39 {
40   fx = dchisq(x, df = df) / schisq(x, df = df)
41   return(fx)
42 }
43
44
45
46
47
48 ##### Plot
49 plot.chisq_seq = function(x, df = 1, xlim=c(0, 10), ylim=c(0, 5), func="dchisq")
50 {
51   color=colorPalette(300)
52
53   len_df = length(df) # df 파라메터의 길이
54
55   color_counter = 1
56   color_counter_init = color_counter
57   legend_name = NULL;
58
59
60   if (func=="dchisq") # 수명분포
61   {
62     plot(x, dchisq(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
63     for (i in 1:len_df) ### 파라메터: df
64     {
65       lines(x, dchisq(x, df=df[i]), col=color[color_counter], lwd=2);
66       color_counter = color_counter + 1;
67       legend_name = c(legend_name, paste("df = ", df[i], sep=""))
68     }
69   }
70   else if (func == "pchisq") # 누적분포함수
71   {
72     plot(x, pchisq(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
73     for (i in 1:len_df) ### 파라메터: df
74     {
75       lines(x, pchisq(x, df=df[i]), col=color[color_counter], lwd=2);
76       color_counter = color_counter + 1;
77       legend_name = c(legend_name, paste("df = ", df[i], sep=""))
78     }
79   }
80   else if (func == "schisq") # 생존함수
81   {
82     plot(x, schisq(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
83     for (i in 1:len_df) ### 파라메터: df
84     {
85       lines(x, schisq(x, df=df[i]), col=color[color_counter], lwd=2);
```

```

86         color_counter = color_counter + 1;
87         legend_name = c(legend_name, paste("df = ", df[i], sep=""))
88     }
89 } else if (func == "hchisq") # 위험함수
90 {
91     plot(x, hchisq(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
92     for (i in 1:len_df) ### 파라미터: df
93     {
94         lines(x, hchisq(x, df=df[i]), col=color[color_counter], lwd=2);
95         color_counter = color_counter + 1;
96         legend_name = c(legend_name, paste("df = ", df[i], sep=""))
97     }
98 }
99 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
100 }
101 }
102
103 par(mfrow = c(2, 2))
104 plot.chisq_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dchisq")
105 plot.chisq_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pchisq")
106 plot.chisq_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 1), func="schisq")
107 plot.chisq_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 2), func="hchisq")

```

Y_1, Y_2, \dots, Y_ν 를 표준정규분포를 따르는 서로 독립인 확률변수라고 하자. 그러면 $X = Y_1^2 + Y_2^2 + \dots + Y_\nu^2$ 는 자유도가 ν 인 카이제곱분포를 따른다. 또한, 카이제곱분포는 감마 분포의 특별한 경우로, 척도 모수 $\theta = 2$ 이고 형상 모수가 β 인 카이제곱분포가 된다.

카이제곱분포의 성질은 다음과 같다.

- 카이제곱분포를 따르는 2개 또는 그 이상의 확률변수들의 합은 카이제곱분포를 따른다. 그 때 자유도는 각각 카이제곱분포의 자유도의 합과 같다.
- $T_1 \sim \chi^2(\nu_1)$, $T_2 \sim \chi^2(\nu_2)$ 이며, 서로 독립인 확률변수라고 하면, $\frac{T_1}{\frac{\nu_1}{\nu_2}} \sim F(\nu_1, \nu_2)$ 가 된다.
- T_1, T_2 가 각각 카이제곱분포를 따르며, 서로 독립이라고 하자. 그러면 $\frac{T_1}{T_1+T_2}$ 는 베타분포를 따른다.
- 자유도 ν 의 값이 커지면 커질수록, 카이제곱분포는 평균이 ν , 분산이 2ν 인 정규분포에 근접한다.

14.9 Cosine Distribution

14.9.1 Ordinary Cosine Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{2b} \cos\left(\frac{t-a}{b}\right)$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\frac{1}{2} [1 - \sin\left(\frac{t-a}{b}\right)]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$[b (\sec\left(\frac{t-a}{b}\right) - \tan\left(\frac{t-a}{b}\right))]^{-1}$	IHR
변수		$a - \frac{2\pi}{2} \leq t \leq a + \frac{b\pi}{2}$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 12: Ordinary Cosine 분포함수에 기반한 척도 함수

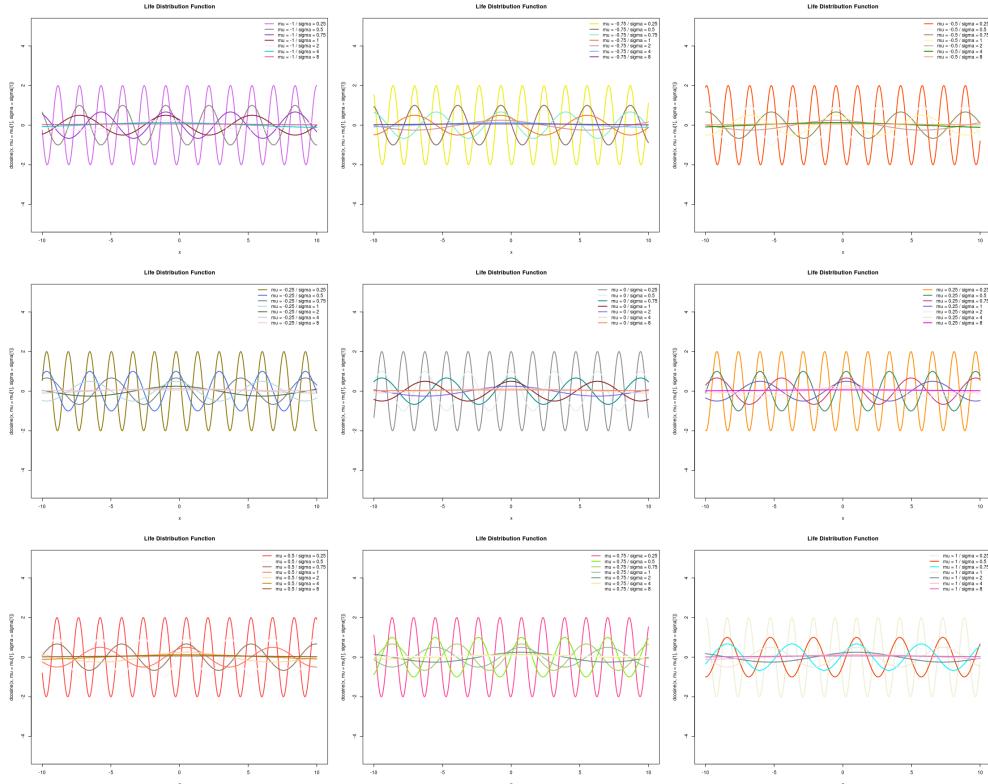


Figure 35: Cosine Distribution에 기반한 수명분포함수

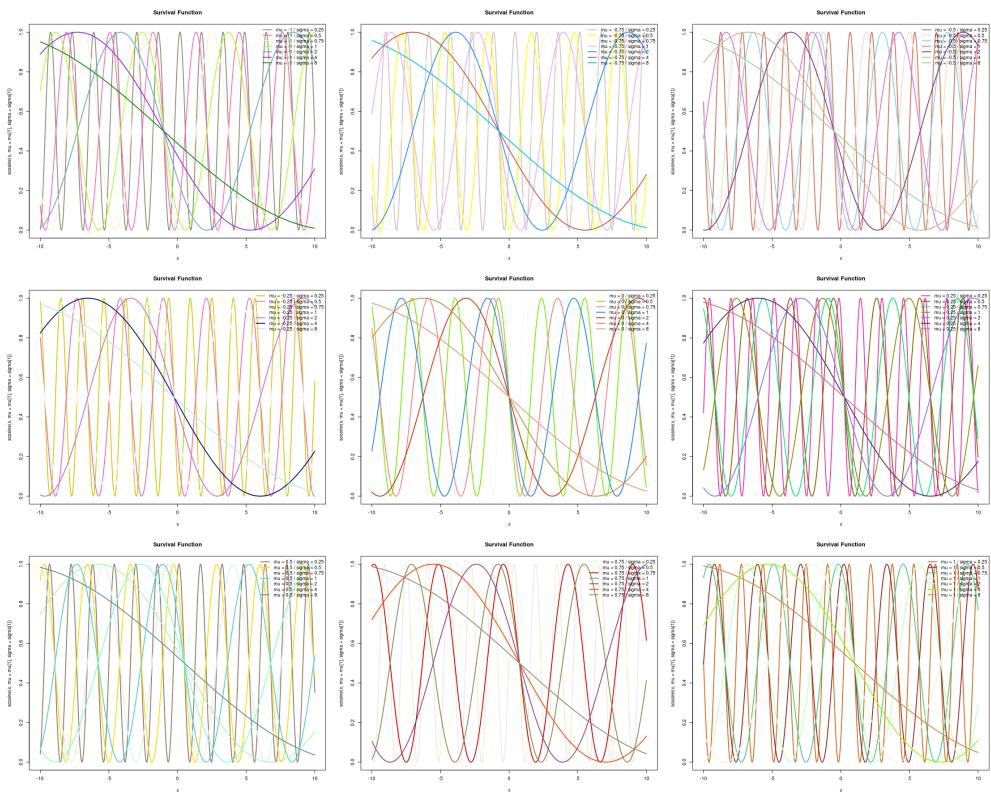


Figure 36: Cosine Distribution에 기반한 생존함수

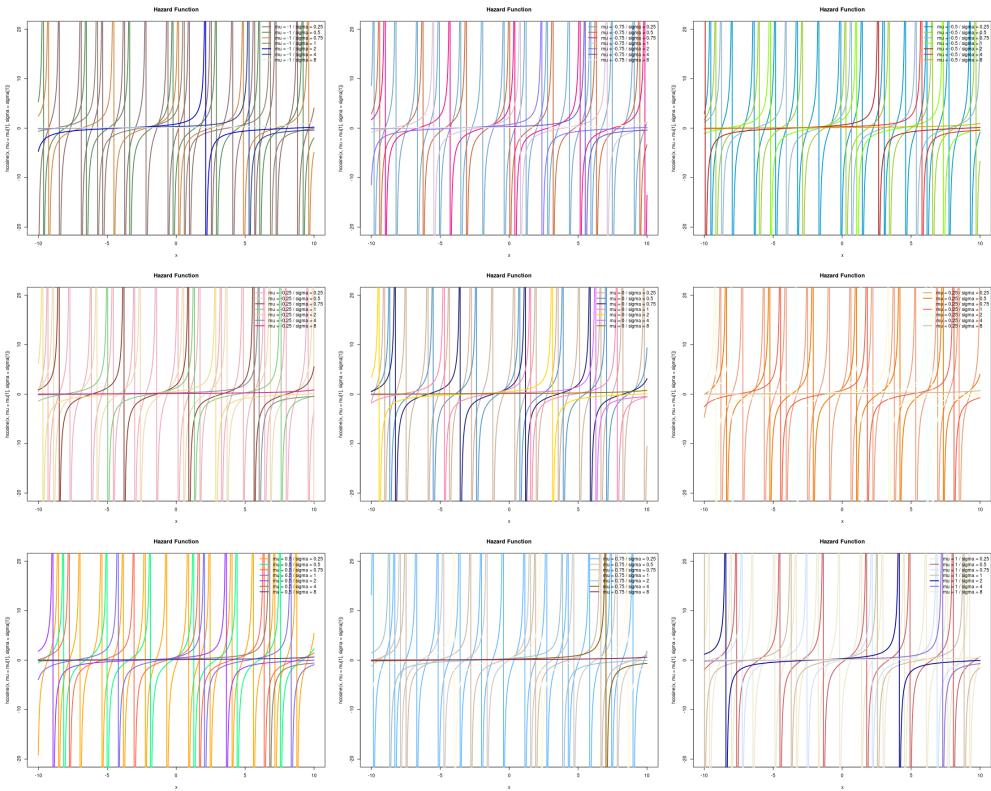


Figure 37: Cosine Distribution에 기반한 위험함수

Code 10. Cosine Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Cosine Distribution
6 ### parameter
7 mu = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 sigma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dcosine = function(x, mu, sigma)
16 {
17   fx = (1/(2*sigma))*(cos((x-mu)/sigma))
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 pcosine = function(x, mu = 1, sigma = 0)
24 {
25   fx = -(scosine(x, mu, sigma) - 1)
26   return(fx)
27 }
28
29
30 ### 생존함수
31 scosine = function (x, mu = 1, sigma = 0)
32 {
33   fx = (1/2) * (1 - sin((x-mu)/sigma))
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hcosine = function (x, mu = 1, sigma = 0)
40 {
41   fx = dcosine(x, mu, sigma) / scosine(x, mu, sigma)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.cosine_seq = function(x, mu = 1, sigma = 0, xlim=c(0, 10), ylim=c(0, 5), func="dcosine")
51 {
52   color=colorPalette(300)
53
54   len_mu = length(mu) # mu 파라메터의 길이
55   len_sigma = length(sigma) # sigma 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_mu) ### 파라메터: mu
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dcosine") # 수명분포
64     {
65       plot(x, dcosine(x, mu=mu[i], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
66           Distribution Function")
67       for (j in 1:len_sigma) ### 파라메터: sigma
68       {
69         lines(x, dcosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("mu = ", mu[i], " / sigma = ", sigma[j], sep=""))
72       }
73     }
74     else if (func == "pcosine") # 누적분포함수
75     {
76       plot(x, pcosine(x, mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
77           Distribution Function")
78       for (j in 1:len_sigma) ### 파라메터: sigma
79       {
80         lines(x, pcosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("mu = ", mu[i], " / sigma = ", sigma[j], sep=""))
83     }
84     else if (func == "scosine") # 생존함수
85     {
86       plot(x, scosine(x, mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival
87           Function")

```

```

86     for (j in 1:len_sigma) ### 파라메터: sigma
87     {
88       lines(x, scosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("mu = ", mu[i], " / sigma = ", sigma[j], sep=""))
91     }
92   }
93 else if (func == "hcosine") # 위험함수
94 {
95   plot(x, hcosine(x, mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
96 Function")
97   for (j in 1:len_sigma) ### 파라메터: sigma
98   {
99     lines(x, hcosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("mu = ", mu[i], " / sigma = ", sigma[j], sep=""))
102  }
103 legend('topright', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.cosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="dcosine")
109
110 par(mfrow = c(3, 3))
111 plot.cosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pcosine")
112
113 par(mfrow = c(3, 3))
114 plot.cosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(0, 1), func="scosine")
115
116 par(mfrow = c(3, 3))
117 plot.cosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(-20, 20), func="hcosine")

```

14.9.2 Raised Cosine Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{2b} [1 + \cos(\pi \frac{t-a}{b})]$	
누적분포함수 (불신뢰도함수)	$F(t) = P(T \leq t)$	작성중	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\frac{1}{2} [1 - \frac{t-a}{b} - \frac{1}{\pi} \sin(\pi \frac{t-a}{b})]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1+\cos(\pi \frac{t-a}{b})}{b[1-\frac{t-a}{b}-\frac{1}{\pi}\sin(\pi \frac{t-a}{b})]}$	IHR
누적 위험률 함수 (누적 고장률 함수)	$H(t) = -\log S(t)$	작성중	
평균수명	$E[T] = \int_0^\infty tf(t)dt$ $= \int_0^\infty S(t)dt$	작성중	
평균잔여수명함수	$m(t) = E[T-t T>t]$ $= \frac{\int_t^\infty S(u)du}{S(t)}$	$b^{0.199339 - \frac{t-a}{2b} + (\frac{t-a}{b})^2 - 0.0506606 \cos(\pi \frac{t-a}{b})}$ $\frac{1}{2}[1 - \frac{t-a}{b} - \frac{1}{\pi} \sin(\pi \frac{t-a}{b})]$	DMRL
k 차 적률	$E(T^k)$	작성중	
100p 백분위수	t_p	작성중	
변수		$a - b \leq t \leq a + b$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 13: Raised Cosine 분포함수에 기반한 척도 함수

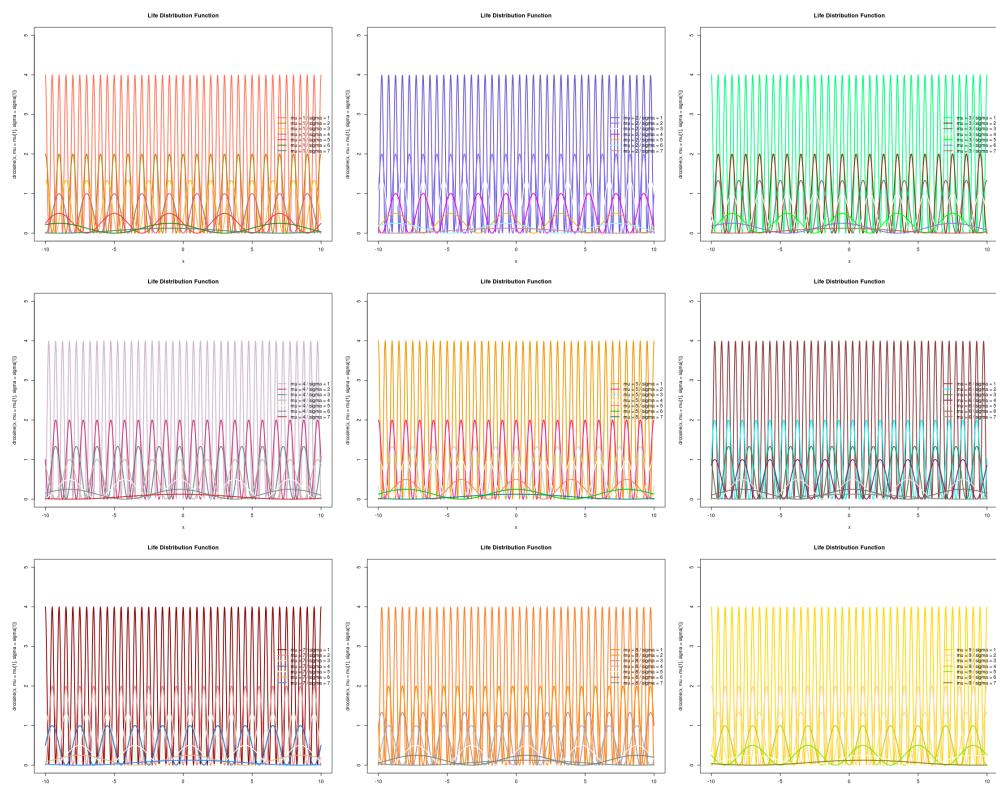


Figure 38: Raised Cosine Distribution에 기반한 수명분포함수

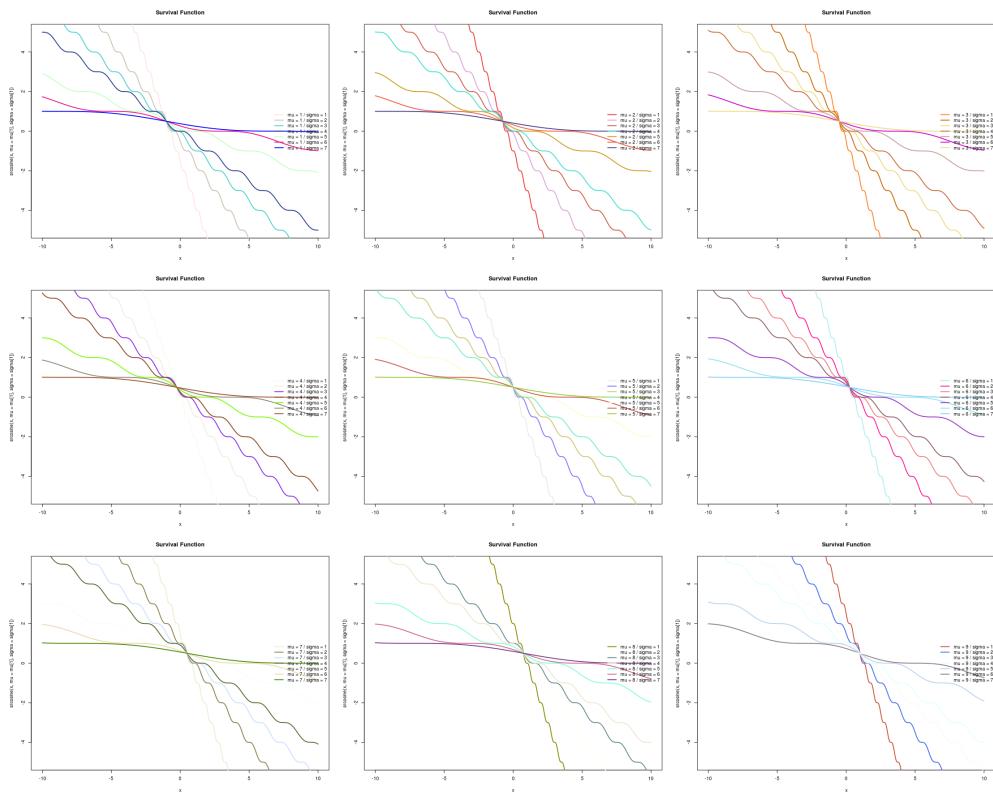


Figure 39: Raised Cosine Distribution에 기반한 생존함수

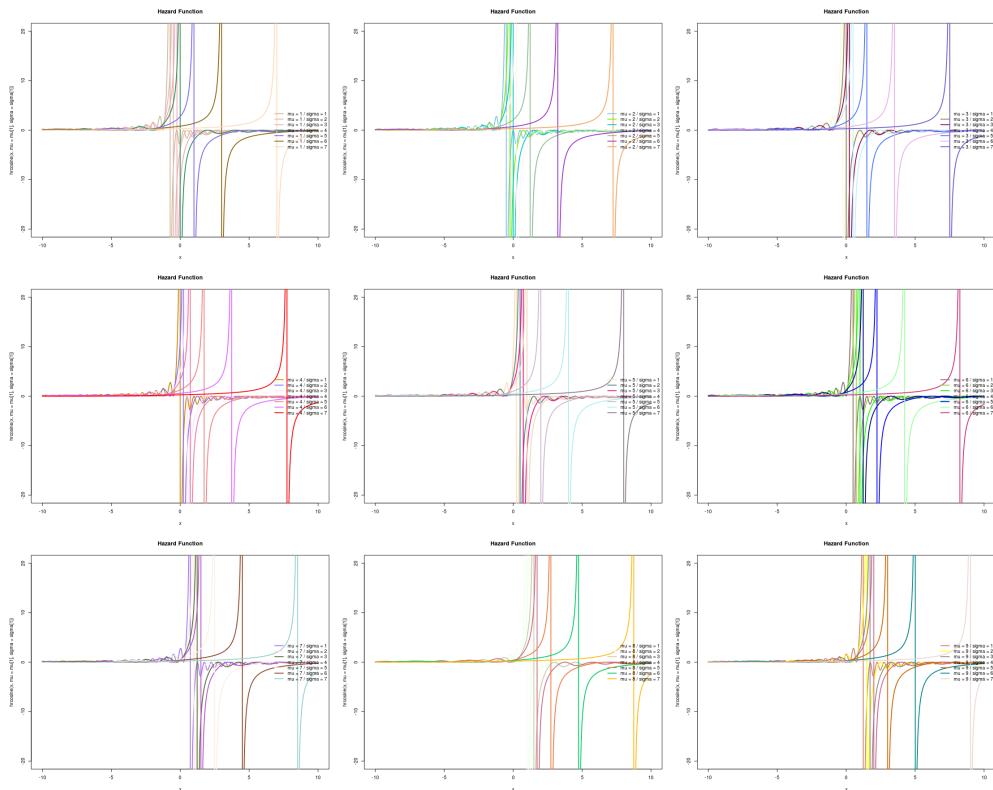


Figure 40: Raised Cosine Distribution에 기반한 위험함수

Code 11. Raised Cosine Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### rcosine Distribution
6 ### parameter
7 mu = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 sigma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 drcosine = function(x, mu, sigma)
16 {
17   fx = (1/(2*sigma))*(1 + cos(pi * (x-mu)/sigma))
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 prcosine = function(x, mu = 1, sigma = 0)
24 {
25   fx = -(srcosine(x, mu, sigma) - 1)
26   return(fx)
27 }
28
29
30 ### 생존함수
31 srcosine = function(x, mu = 1, sigma = 0)
32 {
33   fx = (1/2) * (1 - ((x-mu)/sigma) - (1/pi)*sin(pi * ((x-mu)/sigma)))
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hrcosine = function(x, mu = 1, sigma = 0)
40 {
41   fx = drcosine(x, mu, sigma) / srcosine(x, mu, sigma)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.rcosine_seq = function(x, mu = 1, sigma = 0, xlim=c(0, 10), ylim=c(0, 5), func="drcosine")
51 {
52   color=colorPalette(300)
53
54   len_mu = length(mu) # mu 파라미터의 길이
55   len_sigma = length(sigma) # sigma 파라미터의 길이
56
57   color_counter = 1
58   for (i in 1:len_mu) ### 파라미터: mu
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="drcosine") # 수명분포
64     {
65       plot(x, drcosine(x, mu=mu[i], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
66           Distribution Function")
67       for (j in 1:len_sigma) ### 파라미터: sigma
68       {
69         lines(x, drcosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("mu = ", i, " / sigma = ", j, sep=""))
72       }
73     }
74     else if (func == "prcosine") # 누적분포함수
75     {
76       plot(x, prcosine(x, mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
77           Distribution Function")
78       for (j in 1:len_sigma) ### 파라미터: sigma
79       {
80         lines(x, prcosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("mu = ", i, " / sigma = ", j, sep=""))
83     }
84     else if (func == "srcosine") # 생존함수
85     {
86       plot(x, srcosine(x, mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival
87           Function")
88     }
89   }
90 }
```

```

86     for (j in 1:len_sigma) ### 파라미터: sigma
87     {
88       lines(x, srcosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("mu = ", i, " / sigma = ", j, sep=""))
91     }
92   }
93 else if (func == "hrcosine") # 위험함수
94 {
95   plot(x, hrcosine(x, mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
96   Function")
97   for (j in 1:len_sigma) ### 파라미터: sigma
98   {
99     lines(x, hrcosine(x, mu=mu[i], sigma=sigma[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("mu = ", i, " / sigma = ", j, sep=""))
102  }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.rcosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(0, 5), func="drcosine")
109
110 par(mfrow = c(3, 3))
111 plot.rcosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="prcosine")
112
113 par(mfrow = c(3, 3))
114 plot.rcosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="srcosine")
115
116 par(mfrow = c(3, 3))
117 plot.rcosine_seq(x, mu, sigma, xlim=c(min(x), max(x)), ylim=c(-20, 20), func="hrcosine")

```

14.10 Dhillon's Distribution(딜론 분포)

Dhillon distribution^o라는 명칭이 통용되는 것은 아니지만, Dhillon(1979, 1981)[10]이 만든 분포에 이름이 붙여지지 않았기 때문에 딜론 분포(Dhillon's distribution)라고 부르기로 한다.

$$h(t) = k\lambda ct^{c-1} + (1-k)\beta t^{\beta-1}be^{bt^\beta}$$

^{o]} 때, $0 \leq k \leq 1, \lambda > 0, b > 0, c > 0, \beta > 0$

이 수명분포는 5개의 모수를 가지고 있고, 모수들의 값에 따라 증가, 감소 및 육조 모양의 위험률 함수(고장률 함수)를 가지는 분포이다.

그리고 파라메터의 값에 따라 다음과 같은 관계를 갖기도 한다.

- $c = \beta = 1$: Gompertz-Makeham 분포
- $k = 0, \beta = 1$: 극치 분포(extreme value distribution)
- $k = 1$: Weibull 분포
- $\beta = 0.5$: 육조 모양의 위험률 함수(고장률 함수)

그러나 이 형태가 너무 복잡했기 때문에, Dhillon(1981)[11]을 통해 Dhillon I 분포¹⁸와 Dhillon II 분포¹⁹로 조금 더 간소화, 구체화하였다.

¹⁸Chapter 14.10.1 참고

¹⁹Chapter 14.10.2 참고

14.10.1 Dhillon's I Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(\frac{t-a}{b}\right)^{c-1} \exp\left[1 - e^{\left(\frac{t-a}{b}\right)^c} + \left(\frac{t-a}{b}\right)^c\right]$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\exp\left[1 - e^{\left(\frac{t-a}{b}\right)^c}\right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c}{b} \left(\frac{t-a}{b}\right)^{c-1} e^{\left(\frac{t-a}{b}\right)^c}$	DIHR for $0 < c < 1$ IHR for $c \geq 1$
변수 파라메터		$t \geq a$	
		$a \in \mathbf{R}, b > 0, c > 0$	

Table 14: Dhillon's I Distribution에 기반한 척도 함수



Figure 41: Dhillon's I Distribution에 기반한 수명분포함수

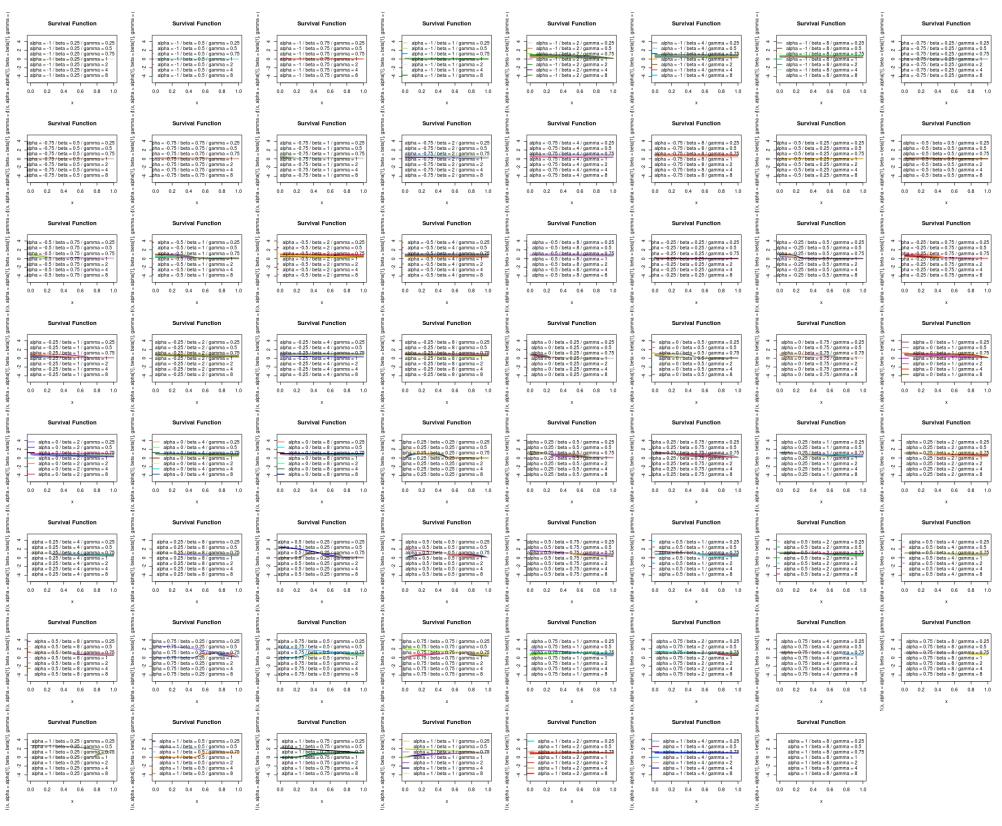


Figure 42: Dhillon's I Distribution에 기반한 생존함수



Figure 43: Dhillon's I Distribution에 기반한 위험함수

Code 12. Dhillon's I Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### dhillon1 Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 1, length.out = 1000)
13
14
15 ### 수명 분포
16 dhillon1 = function(x, alpha = 1, beta = 1, gamma = 1)
17 {
18   fx = (gamma/beta) * ((x - alpha) / beta)^(gamma - 1) * exp(1 - exp(((x - alpha) / beta)^gamma) + ((x - alpha) / beta)^gamma)
19   return(fx)
20 }
21
22
23 ### 누적분포함수
24 pdhillon1 = function(x, alpha = 1, beta = 1, gamma = 1)
25 {
26   fx = -(sdhillon1(x, alpha, beta)) - 1
27   return(fx)
28 }
29
30
31 ### 생존함수
32 sdhillon1 = function (x, alpha = 1, beta = 1, gamma = 1)
33 {
34   fx = exp(1 - exp(((x - alpha) / beta)^gamma))
35   return(fx)
36 }
37
38
39 ### 위험함수
40 hdhillon1 = function (x, alpha = 1, beta = 1, gamma = 1)
41 {
42   fx = dhillon1(x, alpha, beta, gamma) / sdhillon1(x, alpha, beta, gamma)
43   return(fx)
44 }
45
46
47
48
49
50 ##### Plot
51 plot.dhillon1_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="ddhillon1")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라메터의 길이
56   len_beta = length(beta) # beta 파라메터의 길이
57   len_gamma = length(gamma) # gamma 파라메터의 길이
58
59   color_counter = 1
60   for (i in 1:len_alpha) ### 파라메터: alpha
61   {
62     if (func=="ddhillon1") # 수명분포
63     {
64       for (j in 1:len_beta) ### 파라메터: beta
65       {
66         color_counter_init = color_counter
67         legend_name = NULL;
68         plot(x, dhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
69           = 'n', main="Life Distribution Function")
70         for (k in 1:len_gamma) ### 파라메터: gamma
71         {
72           lines(x, dhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
73           color_counter = color_counter + 1;
74           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
75         }
76         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
77       }
78     } else if (func == "pdhillon1") # 누적분포함수
79     {
80       for (j in 1:len_beta) ### 파라메터: beta
81       {
82         color_counter_init = color_counter
83         legend_name = NULL;
84         plot(x, pdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
85           = 'n', main="Cumulative Distribution Function")
86         for (k in 1:len_gamma) ### 파라메터: gamma
87       }
88     }
89   }
90 }
```

```

86  }
87  {
88     lines(x, pdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
89     color_counter = color_counter + 1;
90     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
91   }
92 }
93 }
94 else if (func == "sdhillon1") # 생존함수
95 {
96   for (j in 1:len_beta) ### 파라미터: beta
97   {
98     color_counter_init = color_counter
99     legend_name = NULL;
100    plot(x, sdhillon1(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
101      = 'n', main="Survival Function")
102    for (k in 1:len_gamma) ### 파라미터: gamma
103    {
104      lines(x, sdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
105      color_counter = color_counter + 1;
106      legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
107    }
108  }
109 }
110 else if (func == "hdhillon1") # 위험함수
111 {
112   for (j in 1:len_beta) ### 파라미터: beta
113   {
114     color_counter_init = color_counter
115     legend_name = NULL;
116     plot(x, hdhillon1(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
117       = 'n', main="Hazard Function")
118     for (k in 1:len_gamma) ### 파라미터: gamma
119     {
120       lines(x, hdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
121       color_counter = color_counter + 1;
122       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
123     }
124   }
125 }
126 }
127 }
128
129 par(mfrow = c(8, 8))
130 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="ddhillon1")
131
132 par(mfrow = c(8, 8))
133 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pdhillon1")
134
135 par(mfrow = c(8, 8))
136 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sdhillon1")
137
138 par(mfrow = c(8, 8))
139 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hdhillon1")

```

14.10.2 Dhillon's II Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c+1}{t-a+b} \left[\ln \left(\frac{t-a}{b} + 1 \right) \right]^c e^{-\left[\ln \left(\frac{t-a}{b} + 1 \right) \right]^{c+1}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{-\left[\ln \left(\frac{t-a}{b} + 1 \right) \right]^{c+1}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c+1}{t-a+b} \left[\ln \left(\frac{t-a}{b} + 1 \right) \right]^c$	DHR for $c = 0$ IDHR for $c > 0$
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b \geq 0, c \geq 0$	

Table 15: Dhillon's II Distribution에 기반한 척도 함수



Figure 44: Dhillon's II Distribution에 기반한 수명분포함수



Figure 45: Dhillon's II Distribution에 기반한 생존함수



Figure 46: Dhillon's II Distribution에 기반한 위험함수

Code 13. Dhillon's II Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### dhillon1 Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 1, length.out = 1000)
13
14
15 ### 수명 분포
16 dhillon1 = function(x, alpha = 1, beta = 1, gamma = 1)
17 {
18   fx = (gamma/beta) * ((x - alpha) / beta)^(gamma - 1) * exp(1 - exp(((x - alpha) / beta)^gamma) + ((x - alpha) / beta)^gamma)
19   return(fx)
20 }
21
22
23 ### 누적분포함수
24 pdhillon1 = function(x, alpha = 1, beta = 1, gamma = 1)
25 {
26   fx = -(sdhillon1(x, alpha, beta)) - 1
27   return(fx)
28 }
29
30
31 ### 생존함수
32 sdhillon1 = function (x, alpha = 1, beta = 1, gamma = 1)
33 {
34   fx = exp(1 - exp(((x - alpha) / beta)^gamma))
35   return(fx)
36 }
37
38
39 ### 위험함수
40 hdhillon1 = function (x, alpha = 1, beta = 1, gamma = 1)
41 {
42   fx = dhillon1(x, alpha, beta, gamma) / sdhillon1(x, alpha, beta, gamma)
43   return(fx)
44 }
45
46
47
48
49
50 ##### Plot
51 plot.dhillon1_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="ddhillon1")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라메터의 길이
56   len_beta = length(beta) # beta 파라메터의 길이
57   len_gamma = length(gamma) # gamma 파라메터의 길이
58
59   color_counter = 1
60   for (i in 1:len_alpha) ### 파라메터: alpha
61   {
62     if (func=="ddhillon1") # 수명분포
63     {
64       for (j in 1:len_beta) ### 파라메터: beta
65       {
66         color_counter_init = color_counter
67         legend_name = NULL;
68         plot(x, dhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
69           = 'n', main="Life Distribution Function")
70         for (k in 1:len_gamma) ### 파라메터: gamma
71         {
72           lines(x, dhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
73           color_counter = color_counter + 1;
74           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
75         }
76         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
77       }
78     } else if (func == "pdhillon1") # 누적분포함수
79     {
80       for (j in 1:len_beta) ### 파라메터: beta
81       {
82         color_counter_init = color_counter
83         legend_name = NULL;
84         plot(x, pdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
85           = 'n', main="Cumulative Distribution Function")
86         for (k in 1:len_gamma) ### 파라메터: gamma
87       }
88     }
89   }
90 }
```

```

86  }
87  {
88     lines(x, pdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
89     color_counter = color_counter + 1;
90     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
91   }
92 }
93
94 else if (func == "sdhillon1") # 생존함수
95 {
96   for (j in 1:len_beta) ### 파라미터: beta
97   {
98     color_counter_init = color_counter
99     legend_name = NULL;
100    plot(x, sdhillon1(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
101      = 'n', main="Survival Function")
102    for (k in 1:len_gamma) ### 파라미터: gamma
103    {
104      lines(x, sdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
105      color_counter = color_counter + 1;
106      legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
107    }
108  }
109
110 else if (func == "hdhillon1") # 위험함수
111 {
112   for (j in 1:len_beta) ### 파라미터: beta
113   {
114     color_counter_init = color_counter
115     legend_name = NULL;
116     plot(x, hdhillon1(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
117       = 'n', main="Hazard Function")
118     for (k in 1:len_gamma) ### 파라미터: gamma
119     {
120       lines(x, hdhillon1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
121       color_counter = color_counter + 1;
122       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
123     }
124   }
125 }
126
127 }
128
129 par(mfrow = c(8, 8))
130 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="ddhillon1")
131
132 par(mfrow = c(8, 8))
133 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pdhillon1")
134
135 par(mfrow = c(8, 8))
136 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sdhillon1")
137
138 par(mfrow = c(8, 8))
139 plot.dhillon1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hdhillon1")

```

14.11 Exponential Distribution(지수 분포)

척도 함수	기호	내용
수명분포함수	$f(t)$	$\lambda e^{-\lambda t}$
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{-\lambda t}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	λ (t 에 무관하게 상수임)
변수		$t \geq 0$
파라메터		$\lambda = \frac{1}{\beta} > 0$

Table 16: 지수분포함수에 기반한 척도 함수

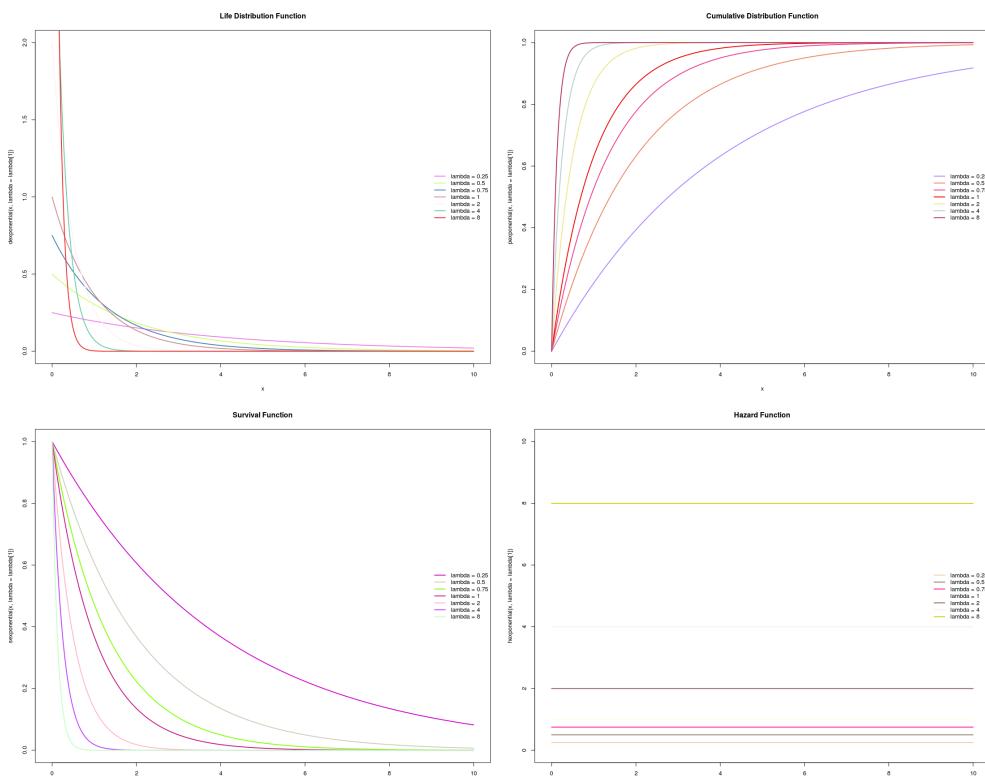


Figure 47: Exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 14. Exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Exponential Distribution
6 ### parameter
7 lambda = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8
9 ### input varialbe
10 x = seq(0, 10, length.out = 1000)
11
12
13 ### 수명 분포
14 dexponential = function(x, lambda = lambda)
15 {
16   fx = lambda * exp(-lambda * x)
17   return(fx)
18 }
19
20
21 ### 누적분포함수
22 pexponential = function(x, lambda = lambda)
23 {
24   fx = 1 - exp(-lambda * x)
25   return(fx)
26 }
27
28
29 ### 생존함수
30 sexponential = function (x, lambda = 1)
31 {
32   fx = exp(-lambda * x)
33   return(fx)
34 }
35
36
37 ### 위험함수
38 hexponential = function (x, lambda = 1)
39 {
40   fx = rep(lambda, length(x))
41   return(fx)
42 }
43
44
45
46
47
48 ##### Plot
49 plot.exponential_seq = function(x, lambda = 1, xlim=c(0, 10), ylim=c(0, 5), func="dexponential")
50 {
51   color=colorPalette(300)
52
53   len_lambda = length(lambda) # lambda 파라미터의 길이
54
55   color_counter = 1
56   color_counter_init = color_counter
57   legend_name = NULL;
58
59
60   if (func=="dexponential") # 수명분포
61   {
62     plot(x, dexponential(x, lambda=lambda[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution
63     Function")
64     for (i in 1:len_lambda) ### 파라미터: lambda
65     {
66       lines(x, dexponential(x, lambda=lambda[i]), col=color[color_counter], lwd=2);
67       color_counter = color_counter + 1;
68       legend_name = c(legend_name, paste("lambda = ", lambda[i], sep=""))
69     }
70   } else if (func == "pexponential") # 누적분포함수
71   {
72     plot(x, pexponential(x, lambda=lambda[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
73     Distribution Function")
74     for (i in 1:len_lambda) ### 파라미터: lambda
75     {
76       lines(x, pexponential(x, lambda=lambda[i]), col=color[color_counter], lwd=2);
77       color_counter = color_counter + 1;
78       legend_name = c(legend_name, paste("lambda = ", lambda[i], sep=""))
79     }
80   } else if (func == "sexponential") # 생존함수
81   {
82     plot(x, sexponential(x, lambda=lambda[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function"
83     )
84     for (i in 1:len_lambda) ### 파라미터: lambda
```

```

84     {
85         lines(x, sexponential(x, lambda=lambda[i]), col=color[color_counter], lwd=2);
86         color_counter = color_counter + 1;
87         legend_name = c(legend_name, paste("lambda = ", lambda[i], sep=""))
88     }
89 }
90 else if (func == "hexponential") # 위험함수
91 {
92     plot(x, hexponential(x, lambda=lambda[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
93     for (i in 1:len_lambda) #### 파라미터: lambda
94     {
95         lines(x, hexponential(x, lambda=lambda[i]), col=color[color_counter], lwd=2);
96         color_counter = color_counter + 1;
97         legend_name = c(legend_name, paste("lambda = ", lambda[i], sep=""))
98     }
99 }
100 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
101 }
102
103 par(mfrow = c(2, 2))
104 plot.exponential_seq(x, lambda, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dexponential")
105 plot.exponential_seq(x, lambda, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pexponential")
106 plot.exponential_seq(x, lambda, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sexponential")
107 plot.exponential_seq(x, lambda, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hexponential")

```

어느 부품의 수명에 대한 위험 함수(고장률 함수)가 상수인 경우를 고려하자. 즉, $h(t) = \lambda$, ($t \geq 0$)이면, 역공식에 의하여 다음과 같은 생존함수 $S(t)$ 를 가진다.

$$S(t) = e^{-\lambda t}$$

위험 함수가 상수인 수명분포는 지수분포가 유일하다. 이러한 이유로 하여금, 지수분포는 비모수 검정에서도 매우 중요한 역할을 한다. 특히 지수분포는 비교적 간단한 형태의 확률밀도함수를 가지고 있으면서, 모수들의 통계적 추론에 필요한 이론적인 결과들이 많이 증명되어 있어, 다른 수명분포에 비하여 광범위한 응용 범위를 가지고 있다. 또한 포아송 과정의 논의에도 필수적인 분포이다.

Note 9. 지수분포에 기반한 k 차 적률

$$\begin{aligned}
 E(T^k) &= \int_0^\infty t^k f(t) dt \\
 &= \int_0^\infty t^k \lambda e^{-\lambda t} dt \\
 &= \frac{1}{\lambda^k} \int_0^\infty x^k e^{-x} dx \quad (\text{substituting } x = \lambda t) \\
 &= \frac{1}{\lambda^k} \Gamma(k+1)
 \end{aligned}$$

지수분포의 기본적인 성질은 다음과 같다.

- 만일 $T \sim \exp(\lambda)$ 이면, $\lambda T \sim \exp(1)$ 이 된다.

Proof: λT 의 생존함수를 $S(t)$ 라고 하면,

$$S(t) = P(\lambda T > t) = P\left(T > \frac{t}{\lambda}\right) = e^{-\lambda \frac{t}{\lambda}} = e^{-t}$$

- 만일 확률변수 $T \sim \text{Unif}(0, 1)$ 이라면, $-\ln T \sim \exp(1)$, $-\ln(1 - T) \sim \exp(1)$ 가 된다.

Proof: $-\ln T$ 의 생존함수를 $S(t)$ 라고 하면,

$$S(t) = P(-\ln T > t) = P(T < e^{-t}) = e^{-t}$$

- 만일 T 가 비음(non-negative)의 연속확률변수이면, $H(T) \sim \exp(1)$ 이 된다.

Proof: $H(T) = \int_0^T h(u)du = \int_0^T \frac{f(u)}{S(u)}du = \int_0^T -\ln S(u)du = -\ln S(T)$ 가 된다.

T 가 비음(non-negative)의 연속확률변수이면, $S(t) \sim \text{Unif}(0, 1)$ 이므로, 위 2번 성질에 의해 $H(T) \sim \exp(1)$ 이 성립 한다.

- (무기억성) 지수분포는 무기억성을 가지는 유일한 수명분포이다.

$$P(T > s + t | T > s) = \frac{P(T > s + t)}{P(T > s)} = \frac{S(s + t)}{S(s)} = \frac{e^{-\lambda(s+t)}}{e^{\lambda s}} = e^{\lambda t} = P(T > t)$$

- 만일 T_1, T_2, \dots, T_n 서로 독립이고, 각각 모수가 $\lambda_1, \lambda_2, \dots, \lambda_n$ 인 지수분포를 가진다면,

$T = \min(T_1, \dots, T_n) \sim \exp(\sum_{i=1}^n \lambda_i)$ 이다.

Proof: T 의 생존함수를 $S(t)$ 라고 하면

$$\begin{aligned} S(t) &= P[\min(T_1, T_2, \dots, T_n) > t] \\ &= P[T_1 > t, T_2 > t, \dots, T_n > t] \\ &= P(T_1 > t)P(T_2 > t) \cdots P(T_n > t) \\ &= e^{-\lambda_1 t}e^{-\lambda_2 t} \cdots e^{-\lambda_n t} \\ &= e^{-\sum_{i=1}^n \lambda_i t} \end{aligned}$$

- $T_i \sim \exp(\lambda)$ ($i = 1, 2, \dots, n$)이고, 서로 독립이라고 하자. $T_{(1)}, T_{(2)}, \dots, T_{(n)}$ 을 T_1, T_2, \dots, T_n 의 순서통계량이라고 하면, $D_k = T_{(k)} - T_{(k-1)}$, ($k = 1, 2, \dots, n$), $T_{(0)} = 0$ 는 k 번째 간격(spacing)으로 정의된다. 그러면

(a) D_1, D_2, \dots, D_n 은 서로 독립이다.

(b) $P(D_k \leq k) = 1 - e^{-(n-k+1)\lambda t}$, ($k = 1, 2, \dots, n$)

(c) $nD_1, (n-1)D_2, \dots, D_n$ 은 표준화 간격(normalized spacing)이라고 부르며, 서로 독립이고, 각각 $\exp(\lambda)$ 인 동일한 분포를 가진다.

- $T_i \sim \exp(\lambda)$ ($i = 1, 2, \dots, n$)이고, 서로 독립이라고 하자. $T_{(r)}$ 을 r 번째 순서통계량이라고 하면,

$$E(T_{(r)}) = \sum_{k=1}^r \frac{1}{(n-k+1)\lambda}$$

$$Var(T_{(r)}) = \sum_{k=1}^r \frac{1}{[(n-k+1)\lambda]^2}$$

Proof: $T_i \sim D_1 + D_2 + \cdots + D_r$ 이거나, 위 5번 성질로부터 $E(D_k) = \frac{1}{n-k+1}$, $Var(D_k) = \frac{1}{(n-k+1)^2}$ 이다.

- $T_i \sim \exp(\lambda)$ ($i = 1, 2, \dots, n$)이고, 서로 독립이라고 하자. 그러면 $2\lambda \sum_{i=1}^n T_i$ 는 자유도가 $2n$ 인 카이제곱 분포를 따른다.

9. Poisson Process와의 관계

X 가 주어진 사건이 단위시간 동안, 또는 단위 면적에서 발생하는 사건의 개수를 나타내는 확률변수라고 하고, 평균 λ 개의 사건이 발생한다고 하자. X 의 확률밀도함수가 다음 $f(x)$ 와 같으면, X 는 모수가 λ 인 포아송 분포를 따른다고 한다.

$$f(x) = P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!} \quad x = 0, 1, \dots$$

(a) $X \sim Poi(\lambda)$ 이면, $E(X) = Var(X) = \lambda$ 이다.

(b) 주어진 사건이 단위시간당 평균 λ 개 발생하는 포아송 분포를 따른다고 하고, 구간 $(0, t]$ ($t > 0$) 동안에 발생하는 사건의 개수를 Y 라고 하자. 그러면 $Y \sim Poi(\lambda t)$ 가 성립한다.

$k = 1, 2, \dots$ 에 대하여, D_k 를 $(k-1)$ 번째와 k 번째 사건의 간격(spacing)을 나타낸다고 하면, D_1, D_2, \dots 는 서로 독립이고, 각각 위험률(고장률)이 λ 인 지수분포를 따른다. $D_k \sim exp(\lambda)$

Proof: 먼저 첫 번째 사건이 발생할 때까지 걸리는 시간인 D_1 의 확률분포를 구해 보자.

D_1 의 생존함수를 $S_1(t)$ 라고 하면, 다음과 같다.

$$\begin{aligned} S_1(t) &= P(D_1 > t) \\ &= P(\text{구간 } (0, t] \text{ 동안에 사건이 0번 발생}) \\ &= \frac{(\lambda t)^0 e^{-\lambda t}}{0!} = e^{-\lambda t} \sim exp(\lambda) \end{aligned}$$

D_2 의 생존함수를 $S_2(t)$ 로 표시하면, 다음과 같다.

$$\begin{aligned} S_2(t) &= P(D_2 > t | D_1 = d_1) \\ &= P(\text{구간 } (d_1, d_1 + t] \text{ 동안에 사건이 0번 발생} | D_1 = d_1) \\ &= P(\text{구간 } (0, t] \text{ 동안에 사건이 0번 발생}) \\ &= e^{-\lambda t} \sim exp(\lambda) \end{aligned}$$

마찬가지 방법으로, 모든 k 에 대하여 $D_k \sim exp(\lambda)$ 이 성립되고, 독립성은 포아송분포의 가정으로부터 성립한다.

(c) T_K 가 $(k-1)$ 번째와 k 번째 사건의 간격(spacing)을 나타내고, T_1, T_2, \dots 가 서로 독립이며, 각각 위험률(고장률)이 λ 인 지수분포를 따른다고 하자. 그러면 구간 $(0, t]$ 동안에 발생하는 사건의 개수는, 모수 λt 를 가지는 포아송 분포를 따른다.

Proof: $T = T_1 + T_2 + \dots + T_n$ ²⁰은, n 번째 사건이 발생할 때까지 걸리는 시간을 나타내며, T 가 다음과 같은 확률밀도함수 $f_T(t)$ 와 생존함수 $S_T(t)$ 를 가진다.

$$f_T(t) = \frac{(\lambda t)^{n-1}}{(n-1)!} \lambda e^{-\lambda t} \quad t \geq 0$$

$$S_T(t) = \sum_{k=0}^{n-1} \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad t \geq 0$$

구간 $(0, T]$ 동안에 발생하는 사건의 개수를 확률변수 N 으로 표시하면,

$$\begin{aligned} P(N = n) &= P[T_1 + T_2 + \dots + T_n \leq t < T_1 + T_2 + \dots + T_n + T_{n+1}] \\ &= P[T_1 + T_2 + \dots + T_n + T_{n+1} > t] - P[T_1 + T_2 + \dots + T_n > t] \\ &= \sum_{k=0}^n \frac{(\lambda t)^k}{k!} e^{-\lambda t} - \sum_{k=0}^{n-1} \frac{(\lambda t)^k}{k!} e^{-\lambda t} \\ &= \frac{(\lambda t)^n}{n!} e^{-\lambda t} \sim Poi(\lambda t) \quad n = 0, 1, 2, \dots \end{aligned}$$

²⁰ 이 T 는 $T \sim Gamma(n, \lambda)$ 를 따른다.

Note 10. 지수분포를 따르는 완전자료에 대한 모수 추정

지수분포는 한 개의 모수를 가지는 수명분포로, 위험률(고장률) λ 또는 평균 수명 $E(T) = \theta = \frac{1}{\lambda}$ 로 표시되는 모수의 값이 주어지면, 지수분포에 대한 모든 특성이 결정된다. 그러나 주어진 수명이 지수분포를 따른다는 것이 알려진 때에도 모수의 값을 모르는 경우가 대부분이며, 따라서 관측된 자료를 이용하여 모수의 값을 추정하여야 한다.

지수분포의 확률밀도함수는

$$f(t|\lambda) = \lambda e^{-\lambda t}, \quad (t \geq 0)$$

또는

$$f(t|\theta) = \frac{1}{\theta} e^{-\frac{t}{\theta}}, \quad (t \geq 0)$$

따라서, 평균수명을 중시할 때에는, θ 를 모수로 하고, 위험률(고장률)을 중시할 때에는 λ 를 모수로 하면 된다.

$T \sim exp(\lambda)$ 를 가정하고, n 개의 부품을 가지고 시점 $t = 0$ 에서 시험을 시작하여, 부품들의 고장 시간이 모두 관측되었다고 가정하자. 이 값들을 t_1, t_2, \dots, t_n 으로 표시하자. 이 경우 얻어진 고장자료는 완전 자료이다.

- 점추정

λ 의 최대 가능도 함수는

$$L(\lambda) = \lambda e^{-t_1} \lambda e^{-t_2} \cdots \lambda e^{-t_n}$$

양변에 로그를 취하면

$$\ln L(\lambda) = n \ln \lambda - \lambda \sum_{i=1}^n t_i$$

양변을 λ 에 미분하고 0으로 놓으면,

$$\frac{\partial L(\lambda)}{\partial \lambda} = \frac{n}{\lambda} - \sum_{i=1}^n t_i = 0$$

이를 λ 에 대해서 풀면, λ 의 최대가능도추정량(MLE)은 다음과 같다.

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^n t_i}$$

이 $\hat{\lambda}$ 는 λ 에 대한 unbiased estimator이며, MLE의 성질에 의해, θ 의 MLE 또한 $\hat{\theta} = \frac{1}{\hat{\lambda}} = \frac{\sum_{i=1}^n t_i}{n}$ 이 된다.

- 구간 추정

λ 에 대한 $100(1 - \alpha)\%$ 신뢰구간을 유도하기 위해서는, 먼저 점 추정량 $\hat{\lambda}$ 의 표본 분포를 구하여야 한다.

$\hat{\lambda}$ 를 확률변수로 간주하기 위하여, $\hat{\lambda} = \frac{n}{\sum_{i=1}^n T_i}$ 를 고려하자. 여기에서 $T_i \sim exp(\lambda)$ ($i = 1, 2, \dots, n$)이고, T_1, T_2, \dots, T_n 은 서로 독립이다. 그러면 **지수분포의 8번 성질**에 의해

$$2\lambda \sum_{i=1}^n T_i = \frac{2n\lambda}{\hat{\lambda}} \sim \chi^2(2n)$$

따라서, $\chi^2(k; p)$ 는, 자유도가 k 인 카이제곱분포의 $100(1 - p)$ 백분위수, 즉, 오른쪽 면적이 p 가 되는 점을 나타낸다고 하면, 다음과 같은 관계가 성립된다.

$$\begin{aligned} 1 - \alpha &= P \left[\chi^2 \left(2n; 1 - \frac{\alpha}{2} \right) < \frac{2n\lambda}{\hat{\lambda}} < \chi^2 \left(2n; \frac{\alpha}{2} \right) \right] \\ &= P \left[\frac{\hat{\lambda}}{2n} \chi^2 \left(2n; 1 - \frac{\alpha}{2} \right) < \lambda < \frac{\hat{\lambda}}{2n} \chi^2 \left(2n; \frac{\alpha}{2} \right) \right] \\ &= P \left[\frac{1}{2 \sum_{i=1}^n T_i} \chi^2 \left(2n; 1 - \frac{\alpha}{2} \right) < \lambda < \frac{1}{2 \sum_{i=1}^n T_i} \chi^2 \left(2n; \frac{\alpha}{2} \right) \right] \end{aligned}$$

$\lambda = \frac{1}{\theta}$ 의 관계를 이용하면, 평균수명 θ 에 대한 $100(1 - \alpha)\%$ 신뢰구간은 다음과 같다.

$$P \left[\frac{2n\hat{\theta}}{\chi^2(2n; 1 - \frac{\alpha}{2})} < \theta < \frac{2n\hat{\theta}}{\chi^2(2n; \frac{\alpha}{2})} \right]$$

대표본²¹인 경우에는, 중심극한정리에 의해, θ 에 대한 대표본 $100(1 - \alpha)\%$ 신뢰구간은 다음과 같이 근사적으로 계산된다.

$$\left[\hat{\theta} - \frac{\hat{\theta}}{\sqrt{n}} z_{\frac{\alpha}{2}}, \hat{\theta} + \frac{\hat{\theta}}{\sqrt{n}} z_{\frac{\alpha}{2}} \right]$$

이는 $E[\hat{\theta}] = \theta$, $Var[\hat{\theta}] = \frac{\theta^2}{n}$ 을 이용한다.

²¹ 대략 $n \geq 30$

Note 11. 지수분포를 따르는 Type II 우중도절단자료(정수중단자료)에 대한 모수 추정

Type II 우중도절단 자료(정수중단자료)는, n 개의 부품을 가지고 $t = 0$ 에서 수명시험을 시작하여, 고장(재발)시간이 순차적으로 기록되고, 미리 정해진 $r (\leq n)$ 개의 부품이 고장나는(환자의 병이 재발하는) 시점에서 시험을 중단하였을 때 얻어지는 자료이다. 이 때 고장(재발)이 관측된 r 개의 부품에 대한 고장(재발) 시간을 순서대로 나열하면, $t_{(1)} \leq t_{(2)} \leq \dots \leq t_{(r)}$ 이 되고, 나머지 $(n - r)$ 개의 부품(환자)는 수명이 $t_{(r)}$ 보다 크다는 것이 기록된다. 이 경우 r 의 값은 고정된 알려진 상수이나, 시험중단시점인 $t_{(r)}$ 의 값은, 부품(환자)의 수명 분포에 의존하는 확률변수이다. Type II 우중도절단 자료(정수중단자료)에 대한 평균수명 θ 에 대한 가능도함수는 다음과 같이 표시된다.

$$\begin{aligned} L(\theta) &= \frac{n!}{(n-r)!} \prod_{i=1}^r \frac{1}{\theta} e^{-\frac{t_{(i)}}{\theta}} e^{-\frac{t_{(i)}}{\theta(n-r)}} \\ &= \frac{n!}{(n-r)!} \frac{1}{\theta^r} e^{-\frac{1}{\theta} [\sum_{i=1}^r t_{(i)} + (n-r)t_{(r)}]} \end{aligned}$$

이 가능성도 함수로부터 구해진 θ 의 평균잔여수명함수는 다음과 같이 얻어진다.

$$\hat{\theta} = \frac{\sum_{i=1}^r t_{(i)} + (n-r)t_{(r)}}{r} = \frac{T_c}{r}$$

여기서, $T_c = \sum_{i=1}^r t_{(i)} + (n-r)t_{(r)}$: 수명시험에 소요된 전체 시험시간

Type II 우중도절단 자료(정수중단자료)의 경우, 평균수명 θ 에 대한 $100(1 - \alpha)\%$ 신뢰구간을 구하기 위해서는, 우선 전체 시험시간 T_c 의 표본 분포를 구해야 한다.

$$\begin{aligned} T_c &= \sum_{i=1}^r t_{(i)} + (n-r)t_{(r)} \\ &= \sum_{i=1}^{r-1} t_{(i)} + (n-r+1)t_{(r)} \\ &= \sum_{i=1}^r (n-i+1)(t_{(i)} - t_{(i-1)}) \\ &= \sum_{i=1}^r (n-i+1)D_i \end{aligned}$$

지수분포의 6번-(c) 성질에 의해 $(n-i+1)D_i \sim \exp(\theta)$ ($i = 1, 2, \dots, r$)이므로, 지수분포의 8번 성질에 의해 ($\lambda = \frac{1}{\theta}$), $2\lambda T_c = \frac{2}{\theta} T_c \sim \chi^2(2r)$ 된다. 따라서, θ 에 대한 $100(1 - \alpha)\%$ 신뢰구간은 다음과 같다.

$$\left[\frac{2T_c}{\chi^2(2r; \frac{\alpha}{2})} < \theta < \frac{2T_c}{\chi^2(2r; 1 - \frac{\alpha}{2})} \right]$$

위험률(고장률) λ 에 대한 $100(1 - \alpha)\%$ 신뢰구간은 $\lambda = \frac{1}{\theta}$ 의 관계를 이용하면, 다음과 같다.

$$\left[\frac{\chi^2(2r; \frac{\alpha}{2})}{2T_c} < \lambda < \frac{\chi^2(2r; 1 - \frac{\alpha}{2})}{2T_c} \right]$$

만일, 대표본²²이면, $\hat{\theta}$ 은 근사적으로 평균이 θ 이고, 다음과 같은 분산을 가진 정규분포를 따른다고 알려져 있다.

$$\widehat{Var}(\hat{\theta}) = \frac{\hat{\theta}^2}{\sum_{i=1}^n \left(1 - e^{-\frac{T_i}{\hat{\theta}}}\right)}$$

T_i 는 i 번째 부품(환자)에 대한 고장(재발)관측시간²³이다. 따라서, Type II 우중도절단 자료(정수중단자료)를 이용한 θ 의 대표본 $100(1 - \alpha)\%$ 신뢰구간은 다음과 같다.

$$\left[\hat{\theta} - z_{\frac{\alpha}{2}} \sqrt{\widehat{Var}(\hat{\theta})} < \theta < \hat{\theta} + z_{\frac{\alpha}{2}} \sqrt{\widehat{Var}(\hat{\theta})} \right]$$

²²대략 $n \geq 30$

²³만일, 시험중단시점까지 고장이 나지 않은 경우에는, 시험중단시간이다.

Note 12. 지수분포를 따르는 Type I 우중도절단자료(정시중단자료)에 대한 모수 추정

Type I 우중도절단 자료(정시중단자료)는 n 개의 부품(환자)을 가지고 시점 $t = 0$ 에서 시작한 수명시험을 미리 정해진 시점 t_c 에서 중단하였을 때 얻어지는 고장(재발)자료로서, 만일 $0 \leq r \leq n$ 개의 고장(재발)이 관측되었다고 가정하고 r 의 고장(재발)시간을 순서대로 나열하면 $t_{(1)} \leq \dots \leq t_{(r)}$ 과 나머지 $(n - r)$ 개의 부품(환자)은 수명이 t_c 보다 크다는 것이 기록된다.

이 경우, 시험중단시점인 t_c 의 값은 고정된 알려진 상수이나, r 은 부품(환자)의 수명 분포에 의존하는 확률변수이다. R 이 시험기간 $(0, t_c]$ 동안에 발생한 고장(재발) 개수를 나타내는 확률변수라고 하면, 정시중단자료에 대한 평균수명 θ 에 대한 가능도함수는 다음과 같이 표시된다.

$$0 < t_{(1)} \leq \dots \leq t_{(r)} < t_c \text{에 대하여}$$

$$\begin{aligned} L(\theta) &= f(t_{(1)}, \dots, t_{(r)} | R = r) f_R(r) \\ &= \left[r! \prod_{i=1}^r \frac{f(t_{(i)})}{F(t_c)} \right] \left[\frac{n!}{r!(n-r)!} \{F(t_c)\}^r \{1 - F(t_c)\}^{n-r} \right] \\ &= \frac{n!}{(n-r)!} \{1 - F(t_c)\}^{n-r} \prod_{i=1}^r f(t_{(i)}) \\ &= \frac{n!}{(n-r)!} \frac{1}{\theta^r} e^{-\frac{1}{\theta} [\sum_{i=1}^r t_{(i)} + (n-r)t_c]} \end{aligned}$$

F 는 부품(환자)의 누적분포함수(불신뢰도함수)로서, $F(t_c) = 1 - e^{-\frac{t_c}{\theta}}$ 이다. 가능도함수를 최대로 하는 θ 의 MLE는 다음과 같이 얻어진다.

$$\hat{\theta} = \frac{\sum_{i=1}^r t_{(i)} + (n-r)t_c}{r} = \frac{T^*}{r}$$

여기서, $T^* = \sum_{i=1}^r t_{(i)} = (n-r)t_c$. 전체 시험시간

$$\frac{2r\hat{\theta}}{\theta} = \frac{2T^*}{\theta} \sim \chi^2(2r)$$

따라서, Type 1 우중도절단 자료(정시중단자료)를 이용한 θ 의 $100(1 - \alpha)\%$ 신뢰구간은 다음과 같다.

$$\left[\frac{2T^*}{\chi^2(2r; \frac{\alpha}{2})} < \theta < \frac{2T^*}{\chi^2(2r; 1 - \frac{\alpha}{2})} \right]$$

14.11.1 Exponential Distribution with Location Parameter

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b}e^{-\frac{t-a}{b}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{-\frac{t-a}{b}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{b}$	IHR and DHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 17: Exponential 분포함수에 기반한 척도 함수

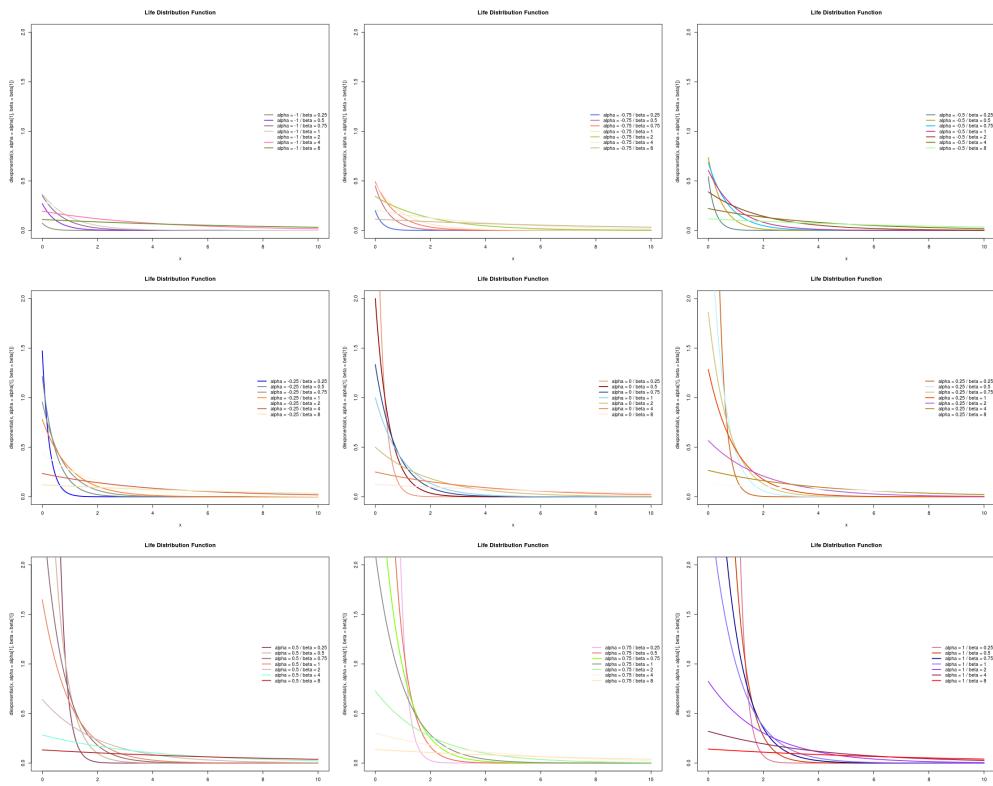


Figure 48: Exponential Distribution에 기반한 수명분포함수

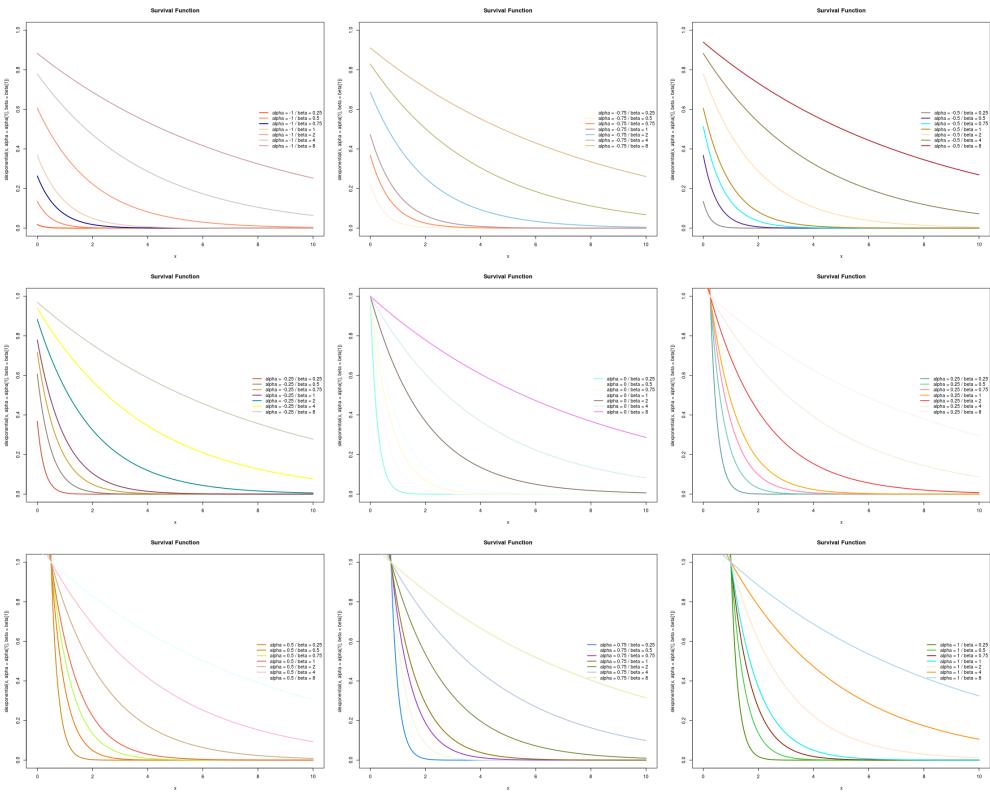


Figure 49: Exponential Distribution에 기반한 생존함수

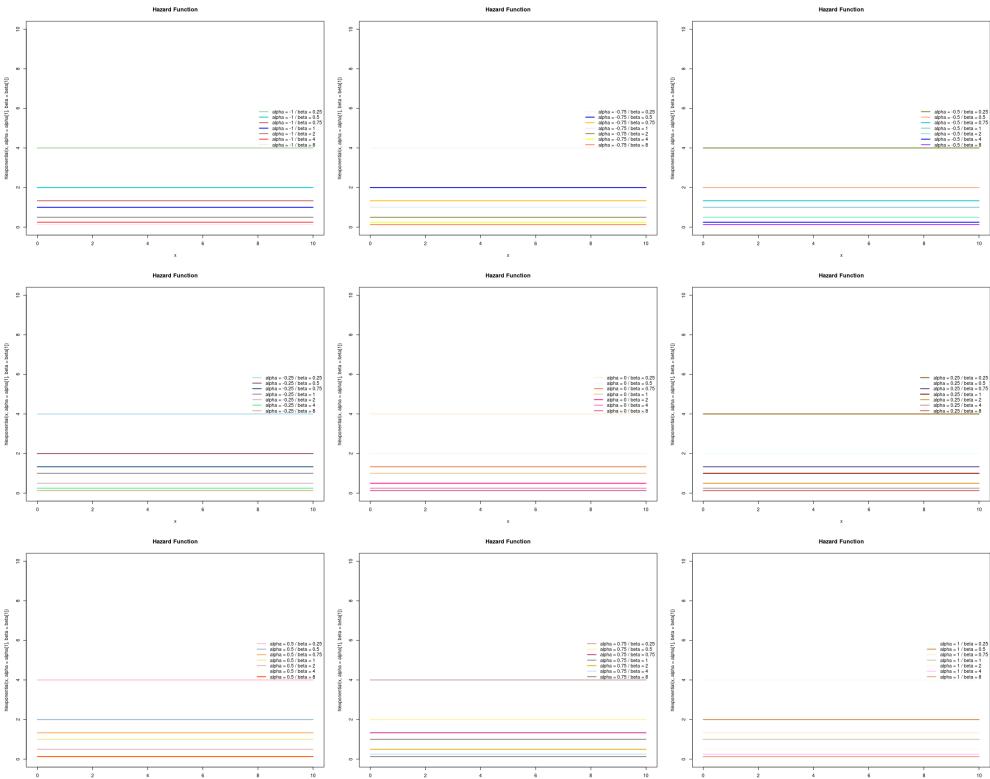


Figure 50: Exponential Distribution에 기반한 위험함수

Code 15. Exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(EnvStats)
4
5 ###### Exponential Distribution with Location Parameter
6 ##### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ##### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ##### 수명 분포
15 dlexponential = function(x, alpha=alpha, beta=beta)
16 {
17   fx = (1/beta) * exp(-((x-alpha)/beta))
18   return(fx)
19 }
20
21
22 ##### 누적분포함수
23 plexponential = function(x, alpha=alpha, beta=beta)
24 {
25   fx = -(slexponential(x, alpha, beta) - 1)
26   return(fx)
27 }
28
29
30
31 ##### 생존함수
32 slexponential = function (x, alpha=alpha, beta=beta)
33 {
34   fx = exp(-((x-alpha)/beta))
35   return(fx)
36 }
37
38
39 ##### 위험함수
40 hlexponential = function (x, alpha=alpha, beta=beta)
41 {
42   fx = dlexponential(x, alpha, beta) / slexponential(x, alpha, beta)
43   return(fx)
44 }
45
46
47
48
49
50 ##### Plot
51 plot.lexponential_seq = function(x, alpha=alpha, beta=beta, xlim=c(0, 10), ylim=c(0, 5), func="dlexponential")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라메터의 길이
56   len_beta = length(beta) # beta 파라메터의 길이
57
58   color_counter = 1
59   for (i in 1:len_alpha) ### 파라메터: alpha
60   {
61     color_counter_init = color_counter
62     legend_name = NULL;
63
64     if (func=="dlexponential") # 수명분포
65     {
66       plot(x, dlexponential(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
67       for (j in 1:len_beta) ### 파라메터: beta
68       {
69         lines(x, dlexponential(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
72       }
73     }
74     else if (func == "plexponential") # 누적분포함수
75     {
76       plot(x, plexponential(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
77       for (j in 1:len_beta) ### 파라메터: beta
78       {
79         lines(x, plexponential(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
80         color_counter = color_counter + 1;
81         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
82       }
83     }
84     else if (func == "slexponential") # 생존함수
85     {
86       plot(x, slexponential(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
87     }
88   }
89 }
```

```

    Survival Function")
87   for (j in 1:len_beta) ### 파라미터: beta
88   {
89     lines(x, slexponential(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
90     color_counter = color_counter + 1;
91     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
92   }
93 }
94 else if (func == "hlexponential") # 위험함수
95 {
96   plot(x, hlexponential(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
97     Hazard Function")
98   for (j in 1:len_beta) ### 파라미터: beta
99   {
100     lines(x, hlexponential(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
101     color_counter = color_counter + 1;
102     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
103   }
104   legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
105 }
106 }
107
108 par(mfrow = c(3, 3))
109 plot.lexponential_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dlexponential")
110
111 par(mfrow = c(3, 3))
112 plot.lexponential_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="plexponential")
113
114 par(mfrow = c(3, 3))
115 plot.lexponential_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="slexponential")
116
117 par(mfrow = c(3, 3))
118 plot.lexponential_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hlexponential")

```

14.11.2 Exponentiated Exponential Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} e^{-\frac{t-a}{b}} \left[1 - e^{-\frac{t-a}{b}}\right]^{c-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \left[1 - e^{-\frac{t-a}{b}}\right]^c$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	DHR for $0 < c \leq 1$ IHR for $c \geq 1$
변수		$t \geq a$	
파라미터		$a \in \mathbf{R}, b > 0$	

Table 18: Exponentiated exponential 분포함수에 기반한 척도 함수

Code 16. Exponentiated exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 ##### Not Completed
2
3 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
4 source("colorPalette.R")
5
6
7 ##### exponentiatedExponential Distribution
8 ### parameter
9 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
10 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
11 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
12
13 ### input varialbe
14 x = seq(0, 10, length.out = 1000)
15
16
17 ### 수명 분포
18 dexponentiatedExponential = function(x, alpha = 1, beta = 1, gamma = 1)
19 {
20   fx = (gamma/beta) * exp(-((x - alpha) / beta)) * (1 - exp(((x - alpha) / beta)))^(gamma-1)
21   return(fx)
22 }
23
24
25 ### 누적분포함수
26 pexponentiatedExponential = function(x, alpha = 1, beta = 1, gamma = 1)
27 {
28   fx = -(sexponentiatedExponential(x, alpha, beta) - 1)
29   return(fx)
30 }
31
32
33 ### 생존함수
34 sexponentiatedExponential = function (x, alpha = 1, beta = 1, gamma = 1)
35 {
36   fx = 1 - (1 - exp(((x - alpha) / beta)))^gamma
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hexponentiatedExponential = function (x, alpha = 1, beta = 1, gamma = 1)
43 {
44   fx = dexponentiatedExponential(x, alpha, beta, gamma) / sexponentiatedExponential(x, alpha, beta, gamma)
45   return(fx)
46 }
47
48
49
50
51
52 ##### Plot
53 plot.exponentiatedExponential_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func=""
54   dexponentiatedExponential")
55 {
56   color=colorPalette(300)

```

```

56
57 len_alpha = length(alpha) # alpha 파라메터의 길이
58 len_beta = length(beta) # beta 파라메터의 길이
59 len_gamma = length(gamma) # gamma 파라메터의 길이
60
61 color_counter = 1
62 for (i in 1:len_alpha) ### 파라메터: alpha
63 {
64   if (func=="dexponentiatedExponential") # 수명분포
65   {
66     for (j in 1:len_beta) ### 파라메터: beta
67     {
68       color_counter_init = color_counter
69       legend_name = NULL;
70       plot(x, dexponentiatedExponential(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color
71           [1], lwd=2, type = 'n', main="Life Distribution Function")
72       for (k in 1:len_gamma) ### 파라메터: gamma
73       {
74         lines(x, dexponentiatedExponential(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter],
75               lwd=2);
76         color_counter = color_counter + 1;
77         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
78       }
79       legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
80     }
81   }
82 else if (func == "pexponentiatedExponential") # 누적분포함수
83 {
84   for (j in 1:len_beta) ### 파라메터: beta
85   {
86     color_counter_init = color_counter
87     legend_name = NULL;
88     plot(x, pexponentiatedExponential(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color
89           [1], lwd=2, type = 'n', main="Cumulative Distribution Function")
90     for (k in 1:len_gamma) ### 파라메터: gamma
91     {
92       lines(x, pexponentiatedExponential(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter],
93             lwd=2);
94       color_counter = color_counter + 1;
95       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
96     }
97     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
98   }
99 else if (func == "sexponentiatedExponential") # 생존함수
100 {
101   for (j in 1:len_beta) ### 파라메터: beta
102   {
103     color_counter_init = color_counter
104     legend_name = NULL;
105     plot(x, sexponentiatedExponential(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color
106           [1], lwd=2, type = 'n', main="Survival Function")
107     for (k in 1:len_gamma) ### 파라메터: gamma
108     {
109       lines(x, sexponentiatedExponential(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter],
110             lwd=2);
111       color_counter = color_counter + 1;
112       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
113     }
114     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
115   }
116 else if (func == "hexponentiatedExponential") # 위험함수
117 {
118   for (j in 1:len_beta) ### 파라메터: beta
119   {
120     color_counter_init = color_counter
121     legend_name = NULL;
122     plot(x, hexponentiatedExponential(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color
123           [1], lwd=2, type = 'n', main="Hazard Function")
124     for (k in 1:len_gamma) ### 파라메터: gamma
125     {
126       lines(x, hexponentiatedExponential(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter],
127             lwd=2);
128       color_counter = color_counter + 1;
129       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
130     }
131   }
132 par(mfrow = c(9, 8))
133 plot.exponentiatedExponential_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dexponentiatedExponential"
)

```

```
134 par(mfrow = c(9, 8))
135 plot.exponentiatedExponential_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="pexponentiatedExponential"
136 )
137 par(mfrow = c(9, 8))
138 plot.exponentiatedExponential_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sexponentiatedExponential")
139
140 par(mfrow = c(9, 8))
141 plot.exponentiatedExponential_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hexponentiatedExponential"
142 )
```

14.11.3 Reflected Exponential Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} e^{\frac{t-a}{b}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - e^{\frac{t-a}{b}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\left[b \left(e^{\frac{t-a}{b}} - 1 \right) \right]^{-1}$	IHR
변수		$t \geq 0$	
파라메터		작성중	

Table 19: Reflected Exponential 분포함수에 기반한 척도 함수

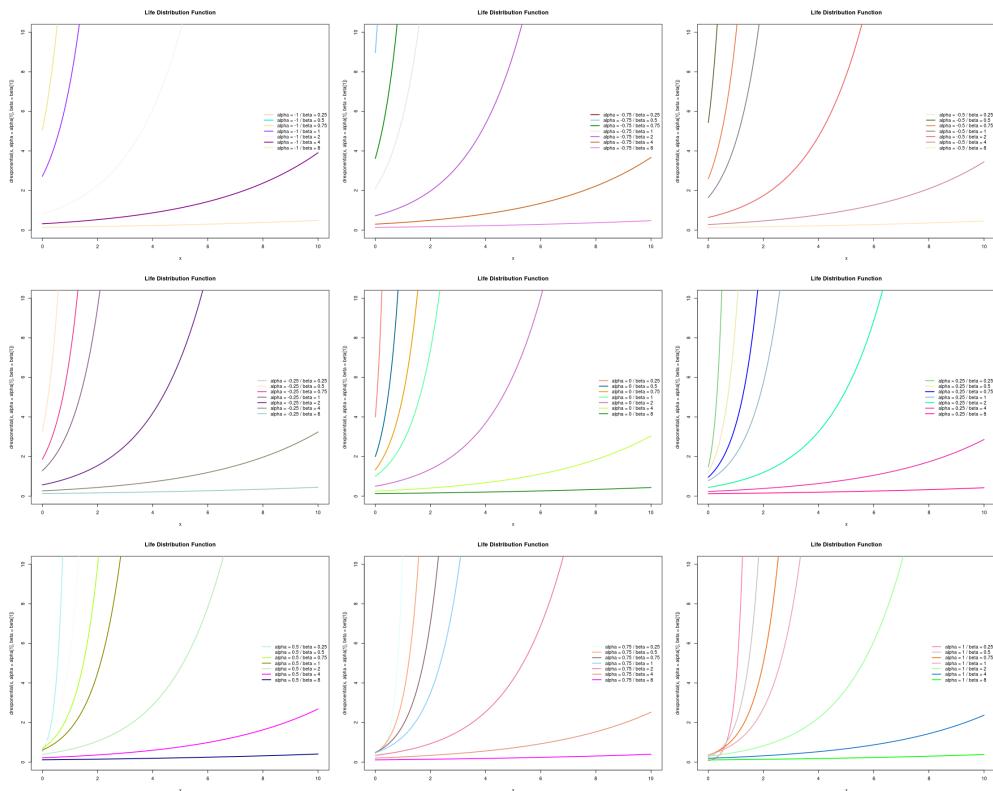


Figure 51: Reflected Exponential Distribution에 기반한 수명분포함수

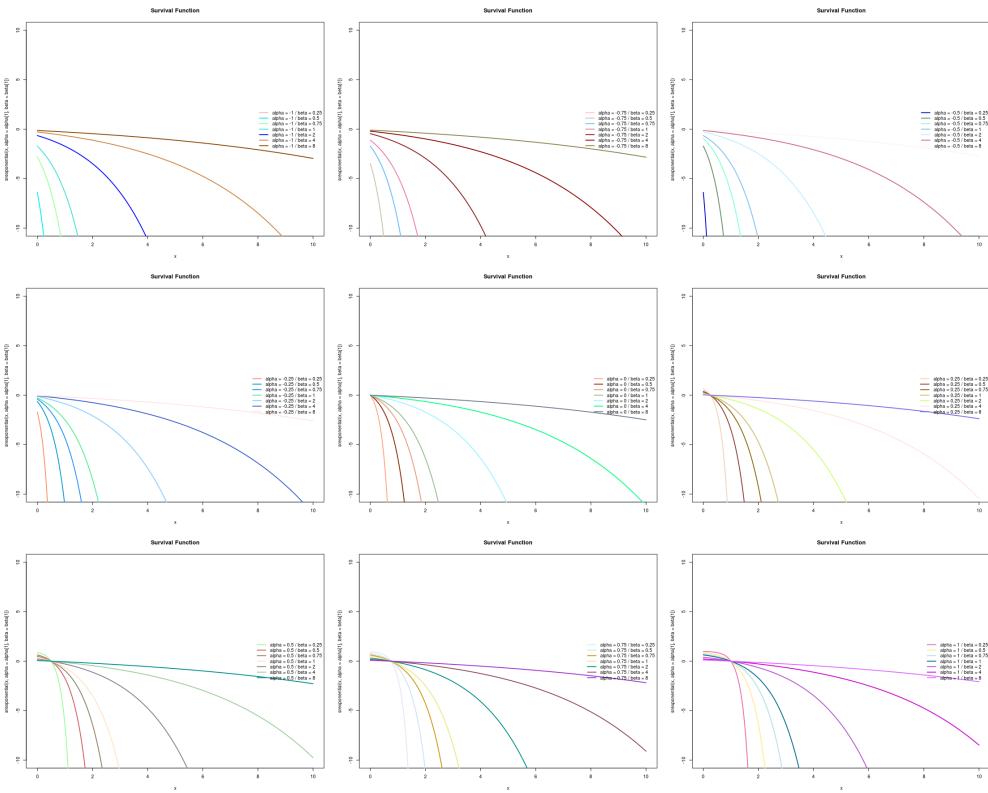


Figure 52: Reflected Exponential Distribution에 기반한 생존함수

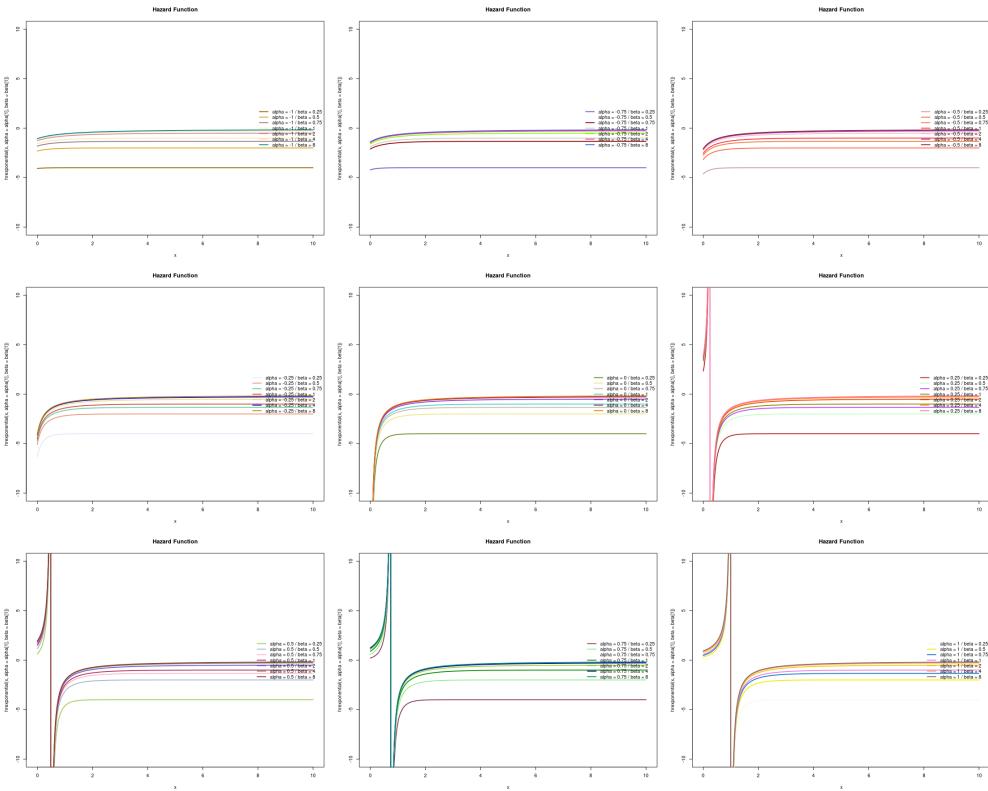


Figure 53: Reflected Exponential Distribution에 기반한 위험함수

Code 17. Reflected Exponential Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Reflected Exponential Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 drexponential = function(x, alpha=alpha, beta=beta)
16 {
17   fx = (1/beta) * exp((x-alpha)/beta)
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 prexponential = function(x, alpha=alpha, beta=beta)
24 {
25   fx = -(srexpontial(x, alpha, beta) - 1)
26   return(fx)
27 }
28
29
30 ### 생존함수
31 srexpontial = function (x, alpha=alpha, beta=beta)
32 {
33   fx = 1 - exp((x-alpha)/beta)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hrexponential = function (x, alpha=alpha, beta=beta)
40 {
41   fx = drexpontial(x, alpha, beta) / srexpontial(x, alpha, beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.rexpontial_seq = function(x, alpha=alpha, beta=beta, xlim=c(0, 10), ylim=c(0, 5), func="drexponential")
51 {
52   color=colorPalette(300)
53
54   len_alpha = length(alpha) # alpha 파라메터의 길이
55   len_beta = length(beta) # beta 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_alpha) ### 파라메터: alpha
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="drexponential") # 수명분포
64     {
65       plot(x, drexpontial(x, alpha=alpha[i], beta=beta[i]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
66       for (j in 1:len_beta) ### 파라메터: beta
67       {
68         lines(x, drexpontial(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
71       }
72     }
73     else if (func == "prexponential") # 누적분포함수
74     {
75       plot(x, prexponential(x, alpha=alpha[i], beta=beta[i]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
76       for (j in 1:len_beta) ### 파라메터: beta
77       {
78         lines(x, prexponential(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
79         color_counter = color_counter + 1;
80         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
81       }
82     }
83     else if (func == "srexpontial") # 생존함수
84     {

```

```

85     plot(x, srexpontial(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
86             Survival Function")
87     for (j in 1:len_beta) ### 파라미터: beta
88     {
89         lines(x, srexpontial(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
90         color_counter = color_counter + 1;
91         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
92     }
93 } else if (func == "hrexponential") # 위험함수
94 {
95     plot(x, hrexponential(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
96             Hazard Function")
97     for (j in 1:len_beta) ### 파라미터: beta
98     {
99         lines(x, hrexponential(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100        color_counter = color_counter + 1;
101        legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102    }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106 par(mfrow = c(3, 3))
107 plot.rexpontial_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="drexponential")
108
109 par(mfrow = c(3, 3))
110 plot.rexpontial_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="prexpontial")
111
112 par(mfrow = c(3, 3))
113 plot.rexpontial_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="srexpontial")
114
115 par(mfrow = c(3, 3))
116 plot.rexpontial_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="hrexponential")
117

```

14.12 Extreme Value Distribution(극치 분포)

14.12.1 최대극치분포(The Maximum Extreme Value Distribution)와 최소극치분포(The Minimum Extreme Value Distribution)

극치 분포는 어떤 분포로부터 무작위로 관찰된 많은 값들의 최소값이나 최대값에 대한 접근 분포(limiting distribution)이다. 수명 자료를 다루는 과정에서 종종 확률변수들의 극단적인 값(최대값이나 최소값)에 관심을 갖게 된다.

- 여러 개의 아이템이 직렬로 구성된 시스템의 수명을 평가할 때, 우리는 모든 아이템의 수명보다는 수명이 가장 짧은 아이템에 더 많은 관심을 갖게 된다. 왜냐하면 이와 같은 **직렬 구조 시스템의 수명은, 수명이 가장 짧은 아이템의 수명과 일치**하기 때문이다.
- 여러 개의 아이템이 병렬로 구성된 시스템의 수명을 평가할 때, 우리는 모든 아이템의 수명보다는 수명이 가장 긴 아이템에 더 많은 관심을 갖게 된다. 왜냐하면 이와 같은 **병렬 구조 시스템의 수명은, 수명이 가장 긴 아이템의 수명과 일치**하기 때문이다.

Note 13. 최대 극치 분포(The Maximum Extreme Value Distribution)

X_1, X_2, \dots, X_n 을 분포함수가 각각 F_1, F_2, \dots, F_n 인 확률변수라고 하자. 만일 Y 가 이 n 개의 확률변수의 최대값이라고 하면, 이것의 분포함수 $F_Y(y)$ 를 최대극치분포라고 부른다. 만일 X_i 들이 동일한 분포인 $F(y)$ 를 따르고 서로 독립이라면, 최대값에 대한 분포함수는 다음과 같이 얻을 수 있다.

$$\begin{aligned} F_Y(y) &= P[(X_1 \leq y)(X_2 \leq y) \cdots (X_n \leq y)] \\ &= \prod_{i=1}^n P(X_i \leq y) \\ &= [F(y)]^n \end{aligned}$$

이 분포함수의 확률밀도함수는 다음과 같이 구해진다.

$$\begin{aligned} f_Y(y) &= \frac{d}{dy} F_Y(y) \\ &= n [F(y)]^{n-1} f(y) \end{aligned}$$

이 경우, 각각의 공통분포인 $F(y)$ 를, 최대극치분포인 $F_Y(y)$ 의 모분포(parent distribution)라고 부른다.

Note 14. 최소 극치 분포(The Minimum Extreme Value Distribution)

X_1, X_2, \dots, X_n 을 분포함수가 각각 F_1, F_2, \dots, F_n 인 확률변수라고 하자. 만일 Z 가 이 n 개의 확률변수의 최소값이라고 하면, 이것의 분포함수 $F_Z(z)$ 를 최소극치분포라고 부른다. 만일 X_i 들이 동일한 분포인 $F(z)$ 를 따르고 서로 독립이라면, 최소값에 대한 분포함수는 다음과 같이 얻을 수 있다.

$$\begin{aligned} F_Z(z) &= P[\min(X_1, X_2, \dots, X_n) \leq z] \\ &= 1 - P[\min(X_1, X_2, \dots, X_n) > z] \\ &= 1 - P(Z > z) \\ &= 1 - \prod_{i=1}^n [1 - F_i(z)] \\ &= 1 - \prod_{i=1}^n [1 - F(z)] \\ &= 1 - [1 - F(z)]^n \end{aligned}$$

여기서, $P(Z > z) = \prod_{i=1}^n P(X_i > z)$: Z 의 생존 함수

이 분포함수의 확률밀도함수는 다음과 같이 구해진다.

$$\begin{aligned} f_Z(z) &= \frac{d}{dz} F_Z(z) \\ &= n [1 - F(z)]^{n-1} f(z) \end{aligned}$$

14.12.2 Type I 최소값 극치분포(Gumbel 최소값 분포)

Table 20: Type I 극치분포(Gumbel 최소값 분포)함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$\frac{1}{\beta} e^{\frac{t-\mu}{\beta}} e^{-e^{\frac{t-\mu}{\beta}}}$
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$e^{-e^{\frac{t-\mu}{\beta}}}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{\beta} e^{\frac{t-\mu}{\beta}}$
파라메터	μ (위치 모수; location parameter), β (척도 모수; scale parameter)	
상수	$\gamma = 0.5772 \dots$ (오일러 상수)	

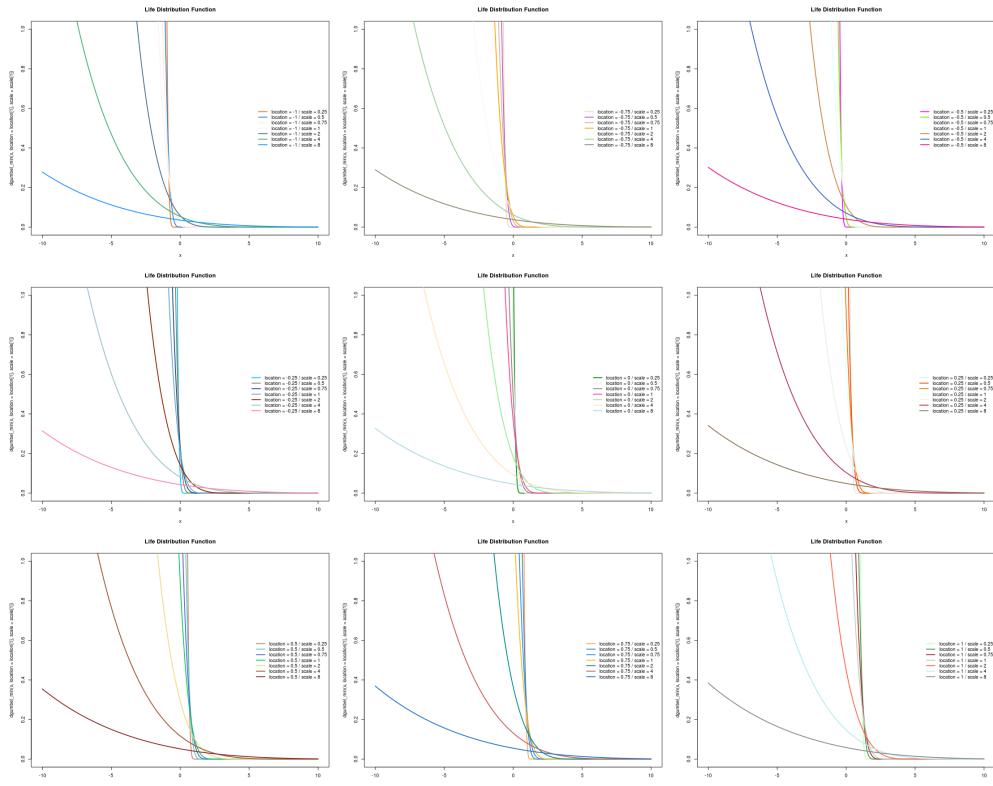


Figure 54: Type I 극치분포(Gumbel 최소값 분포) Distribution에 기반한 수명분포함수

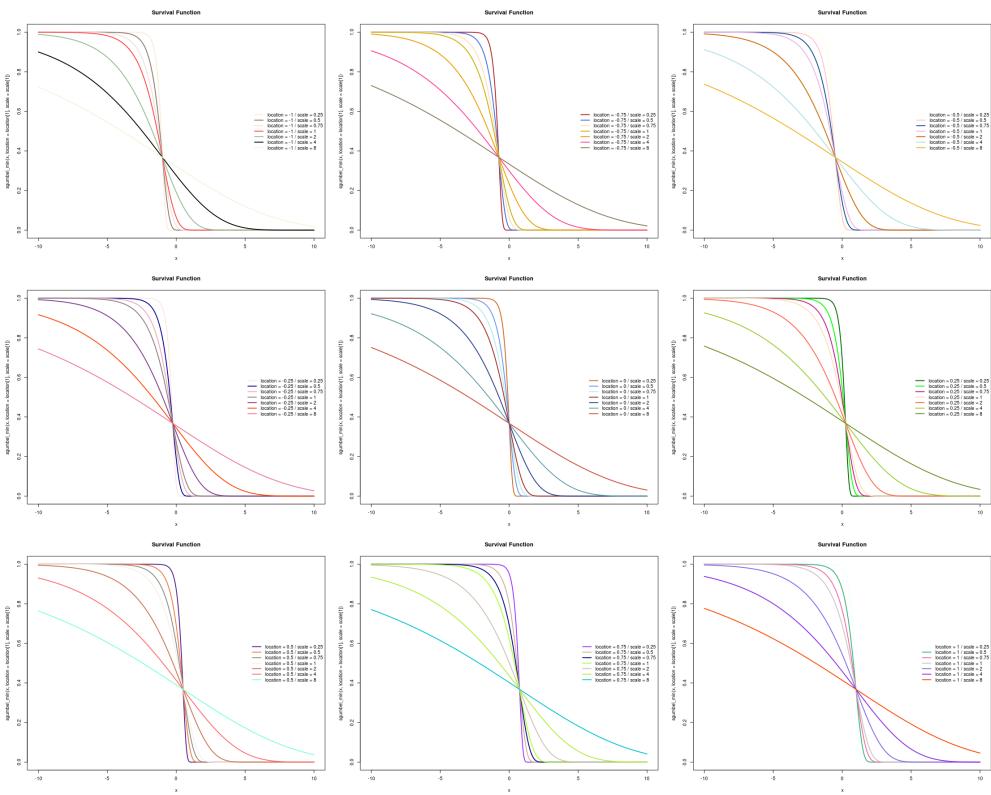


Figure 55: Type I 극치분포(Gumbel 최소값 분포) Distribution에 기반한 생존함수

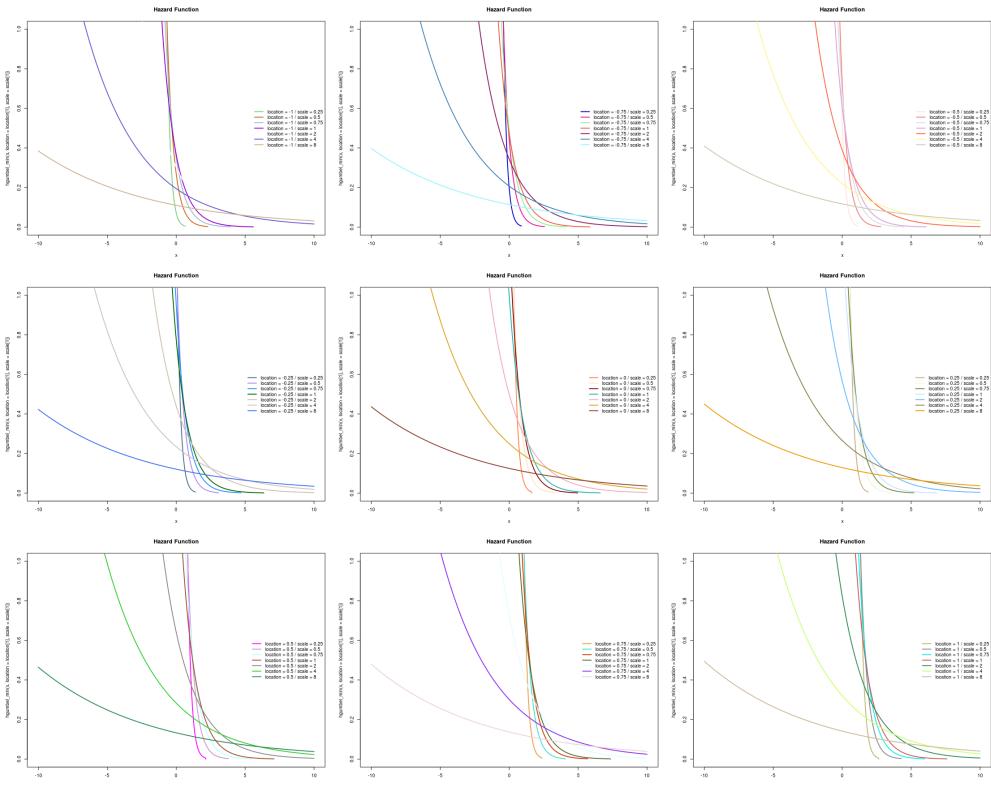


Figure 56: Type I 극치분포(Gumbel 최소값 분포) Distribution에 기반한 위험함수

Code 18. Type I 극치분포(Gumbel 최소값 분포) Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### 극치 분포: Gumbel 최소값 분포
6 ### parameter
7 location = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dgumbel_min = function (x, scale = 1, location = 0)
16 {
17   fx = 1/scale * exp(-(x - location)/scale) * exp(-exp((x - location)/scale))
18   return(fx)
19 }
20
21
22 ### 난수 함수
23 rgumbel_min = function (x, scale = 1, location = 0)
24 {
25   qgumbel_min(runif(x), scale, location)
26 }
27
28
29 ### 누적분포함수
30 pgumbel_min = function (q, scale = 1, location = 0)
31 {
32   fx = -(sgumbel_min(x, scale, location) - 1)
33   return(fx)
34 }
35
36
37 ### 생존함수
38 sgumbel_min = function (x, scale = 1, location = 0)
39 {
40   fx = exp(- exp((x - location)/scale))
41   return(fx)
42 }
43
44
45 ### 위험함수
46 hgumbel_min = function (x, scale = 1, location = 0)
47 {
48   fx = dgumbel_min(x, scale, location) / sgumbel_min(x, scale, location)
49   return(fx)
50 }
51
52
53
54
55
56 ##### Plot
57 plot.gumbel_min_seq = function(x, location = 1, scale = 0, xlim=c(0, 10), ylim=c(0, 5), func="dgumbel_min")
58 {
59   color=colorPalette(300)
60
61   len_location = length(location) # location 파라메터의 길이
62   len_scale = length(scale) # scale 파라메터의 길이
63
64   color_counter = 1
65   for (i in 1:len_location) ### 파라메터: location
66   {
67     color_counter_init = color_counter
68     legend_name = NULL;
69
70     if (func=="dgumbel_min") # 수명분포
71     {
72       plot(x, dgumbel_min(x, location=location[i], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
73             main="Life Distribution Function")
74       for (j in 1:len_scale) ### 파라메터: scale
75       {
76         lines(x, dgumbel_min(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
77         color_counter = color_counter + 1;
78         legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
79       }
80     }
81     else if (func == "pgumbel_min") # 누적분포함수
82     {
83       plot(x, pgumbel_min(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
84             main="Cumulative Distribution Function")
85       for (j in 1:len_scale) ### 파라메터: scale
86       {

```

```

85     lines(x, pgumbel_min(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
86     color_counter = color_counter + 1;
87     legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
88   }
89 }
90 else if (func == "sgumbel_min") # 생존함수
91 {
92   plot(x, sgumbel_min(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
93       main="Survival Function")
94   for (j in 1:len_scale) ### 파라미터: scale
95   {
96     lines(x, sgumbel_min(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
97     color_counter = color_counter + 1;
98     legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
99   }
100 else if (func == "hgumbel_min") # 위험함수
101 {
102   plot(x, hgumbel_min(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
103       main="Hazard Function")
104   for (j in 1:len_scale) ### 파라미터: scale
105   {
106     lines(x, hgumbel_min(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
107     color_counter = color_counter + 1;
108     legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
109   }
110 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
111 }
112 }
113 par(mfrow = c(3, 3))
114 plot.gumbel_min_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="dgumbel_min")
115
116 par(mfrow = c(3, 3))
117 plot.gumbel_min_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pgumbel_min")
118
119 par(mfrow = c(3, 3))
120 plot.gumbel_min_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sgumbel_min")
121
122 par(mfrow = c(3, 3))
123 plot.gumbel_min_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="hgumbel_min")
124

```

Type 1 극치분포는 와이블분포의 대수변환분포로, 위치-척도 모수를 갖는 분포족에 속해서 많이 사용한다.
이 분포는 최소값에 근거한 분포와 최대값에 근거한 분포 두 종류가 있으며, Gumbel 분포로 알려져 있다.

- 최소값에 근거한 점근분포는, 미사일에서 부식성의 화학 물질을 보내는 관의 누출시간을 모형화하거나, 기계 고장 문제를 모형화하는 데 적용된다.
- 최대값에 대한 점근분포는 최대 방류량이나 연중 최고 조류 등을 모형화하는 데 적합하다.

14.12.3 Type I 최대값 극치분포(Gumbel 최대값 분포)

Table 21: Type I 극치분포(Gumbel 최대값 분포)함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$\frac{1}{\beta} e^{-\frac{t-\mu}{\beta}} e^{-e^{-\frac{t-\mu}{\beta}}}$
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - e^{-e^{-\frac{t-\mu}{\beta}}}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{\frac{1}{\beta} e^{-\frac{t-\mu}{\beta}}}{e^{e^{-\frac{t-\mu}{\beta}}} - 1}$
파라미터 상수	$\mu(\text{위치 모수; location parameter}), \beta(\text{척도 모수; scale parameter})$ $\gamma = 0.5772 \dots (\text{오일러 상수})$	

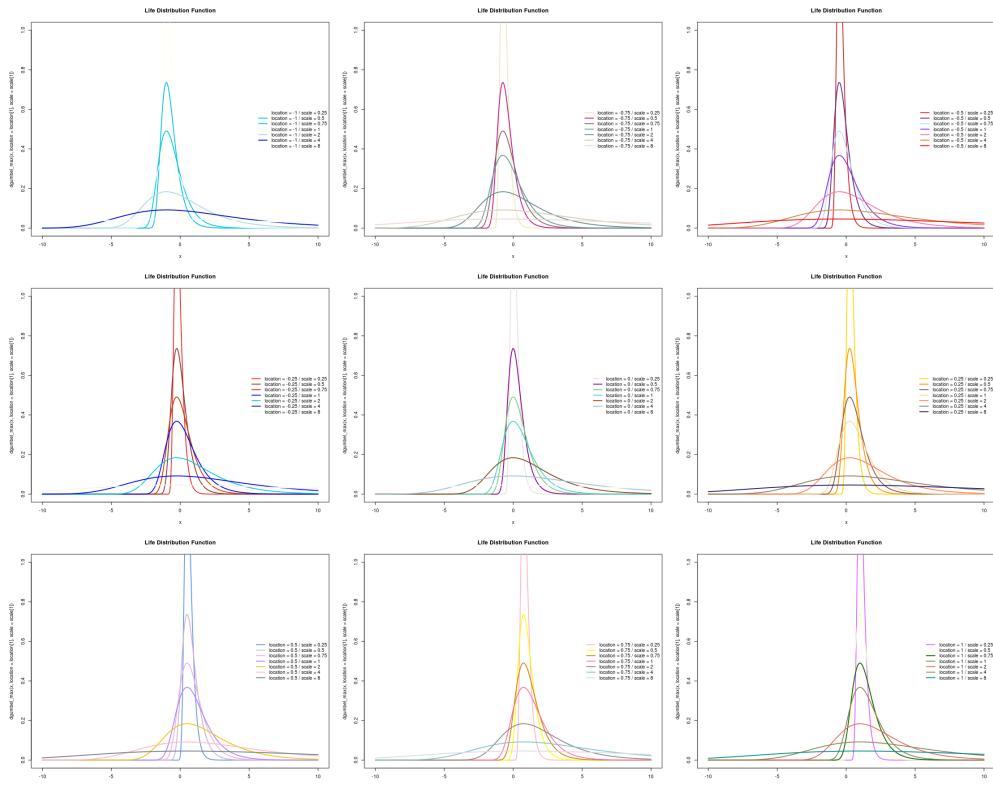


Figure 57: Type I 극치분포(Gumbel 최대값 분포) Distribution에 기반한 수명분포함수

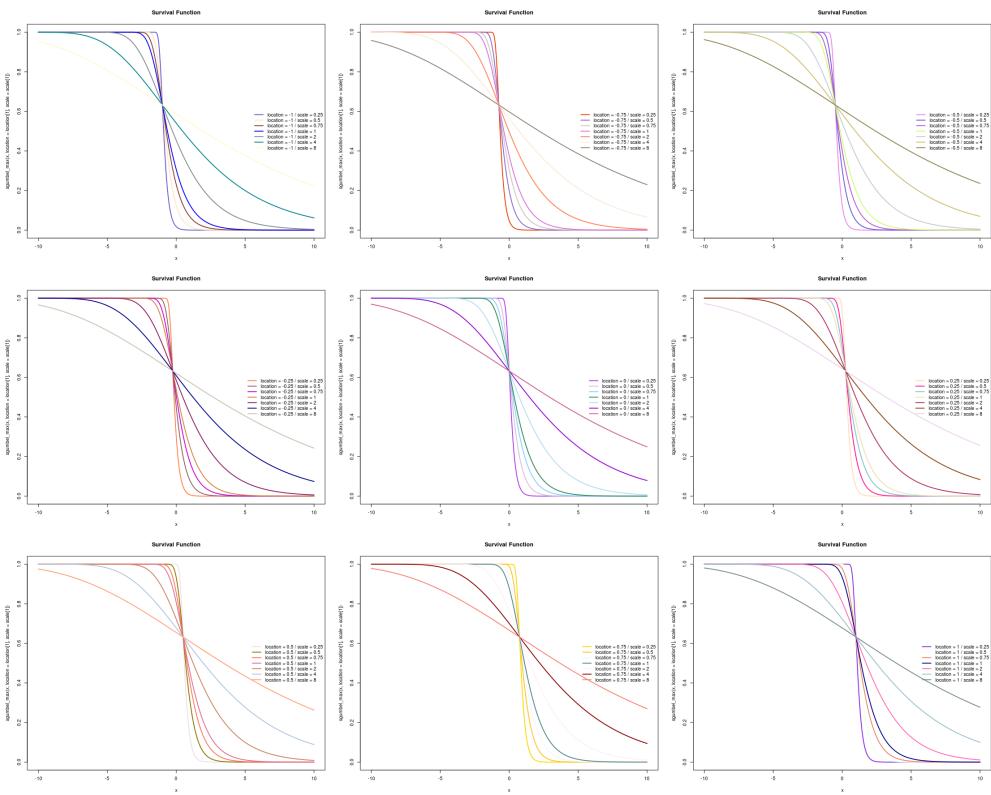


Figure 58: Type I 극치분포(Gumbel 최대값 분포) Distribution에 기반한 생존함수

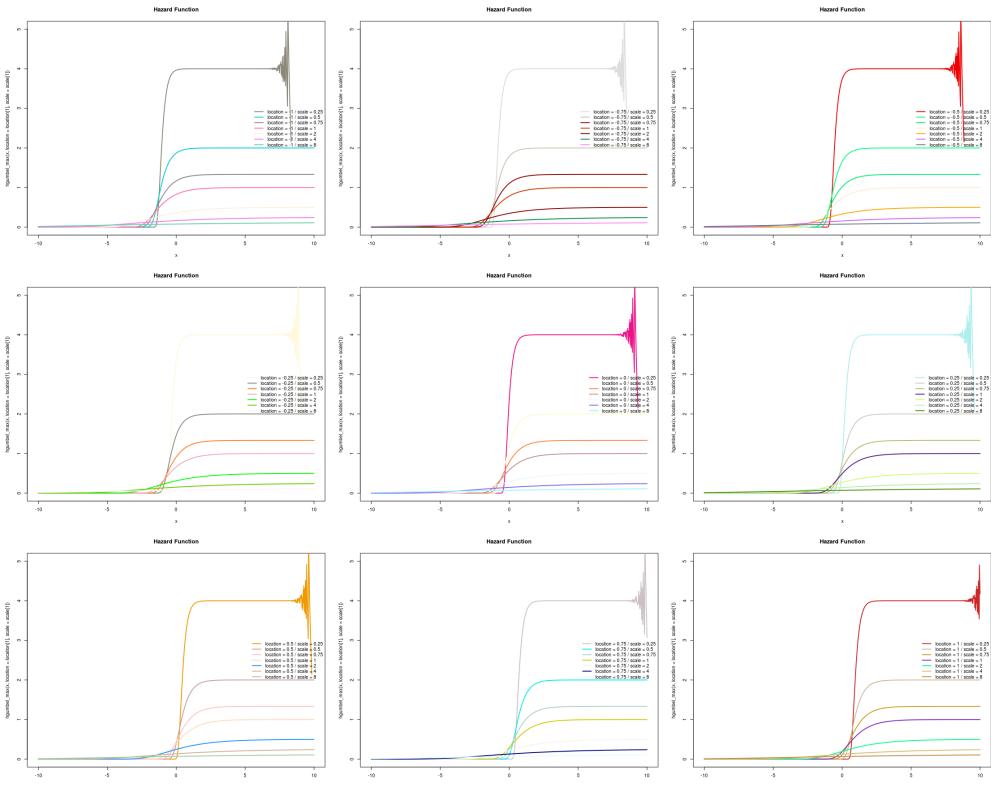


Figure 59: Type I 극치분포(Gumbel 최대값 분포) Distribution에 기반한 위험함수

Code 19. Type I 극치분포(Gumbel 최대값 분포) Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### 극치 분포: Gumbel 최대값 분포
6 ### parameter
7 location = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dgumbel_max = function (x, scale = 1, location = 0)
16 {
17   fx = 1/scale * exp(-(x - location)/scale) * exp(-exp(-(x - location)/scale))
18   return(fx)
19 }
20
21
22 ### 난수 함수
23 rgumbel_max = function (x, scale = 1, location = 0)
24 {
25   qgumbel(runif(n), scale, location)
26 }
27
28
29 ### 누적분포함수
30 pgumbel_max = function (q, scale = 1, location = 0)
31 {
32   fx = -(sgumbel_max(x, scale, location) - 1)
33   return(fx)
34 }
35
36
37 ### 생존함수
38 sgumbel_max = function (x, scale = 1, location = 0)
39 {
40   fx = 1 - exp(-exp(-(x - location)/scale))
41   return(fx)
42 }
43
44
45 ### 위험함수
46 hgumbel_max = function (x, scale = 1, location = 0)
47 {
48   fx = dgumbel_max(x, scale, location) / sgumbel_max(x, scale, location)
49   return(fx)
50 }
51
52
53
54
55
56 ##### Plot
57 plot.gumbel_max_seq = function(x, location = 1, scale = 0, xlim=c(0, 10), ylim=c(0, 5), func="dgumbel_max")
58 {
59   color=colorPalette(300)
60
61   len_location = length(location) # location 파라메터의 길이
62   len_scale = length(scale) # scale 파라메터의 길이
63
64   color_counter = 1
65   for (i in 1:len_location) ### 파라메터: location
66   {
67     color_counter_init = color_counter
68     legend_name = NULL;
69
70     if (func=="dgumbel_max") # 수명분포
71     {
72       plot(x, dgumbel_max(x, location=location[i], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
73           main="Life Distribution Function")
74       for (j in 1:len_scale) ### 파라메터: scale
75       {
76         lines(x, dgumbel_max(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
77         color_counter = color_counter + 1;
78         legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
79       }
80     }
81     else if (func == "pgumbel_max") # 누적분포함수
82     {
83       plot(x, pgumbel_max(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
84           main="Cumulative Distribution Function")
85       for (j in 1:len_scale) ### 파라메터: scale
86     {

```

```

85     lines(x, pgumbel_max(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
86     color_counter = color_counter + 1;
87     legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
88   }
89 }
90 else if (func == "sgumbel_max") # 생존함수
91 {
92   plot(x, sgumbel_max(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
93       main="Survival Function")
94   for (j in 1:len_scale) ### 파라미터: scale
95   {
96     lines(x, sgumbel_max(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
97     color_counter = color_counter + 1;
98     legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
99   }
100 else if (func == "hgumbel_max") # 위험함수
101 {
102   plot(x, hgumbel_max(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n',
103       main="Hazard Function")
104   for (j in 1:len_scale) ### 파라미터: scale
105   {
106     lines(x, hgumbel_max(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
107     color_counter = color_counter + 1;
108     legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
109   }
110 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
111 }
112 }
113 par(mfrow = c(3, 3))
114 plot.gumbel_max_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="dgumbel_max")
115
116 par(mfrow = c(3, 3))
117 plot.gumbel_max_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pgumbel_max")
118
119 par(mfrow = c(3, 3))
120 plot.gumbel_max_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sgumbel_max")
121
122 par(mfrow = c(3, 3))
123 plot.gumbel_max_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 5), func="hgumbel_max")
124

```

14.12.4 Type II 극치분포(Frechet 분포)

Table 22: Type II 극치분포(Frechet 분포)함수에 기반한 척도 함수

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{m}{\beta} \left(\frac{t-\mu}{\beta}\right)^{-(m+1)} e^{-\left(\frac{t-\mu}{\beta}\right)^{-m}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - e^{-\left(\frac{t-\mu}{\beta}\right)^{-(m+1)}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{\frac{m}{\beta} \left(\frac{t-\mu}{\beta}\right)^{-(m+1)}}{e^{\left(\frac{t-\mu}{\beta}\right)^{-m}} - 1}$	
변수		$t \geq 0$	
파라메터		$t \geq \mu, -\infty < \mu < \infty$ (위치 모수; location parameter) $m > 0$ (형상 모수; shape parameter), $\beta > 0$ (척도 모수; scale parameter)	

Code 20. Type II 극치분포(Frechet 최대값 분포)에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(VGAM)
4
5 ###### frechet Distribution
6 ### parameter
7 location = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 shape = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
11
12 ### input varialbe
13 x = seq(0, 10, length.out = 1000)
14
15
16 ### 수명 분포
17 dfrechet(x, location = 0, shape = 1, scale = 1)
18
19
20 ### 분위수 함수
21 qfrechet(x, location = 0, shape = 1, scale = 1)
22
23
24 ### 난수 함수
25 rfrechet(x, location = 0, shape = 1, scale = 1)
26
27
28 ### 누적분포함수
29 pfrechet(x, location = 0, shape = 1, scale = 1)
30
31
32 ### 생존함수
33 sfrechet = function (x, location = 1, shape = 1, scale = 1)
34 {
35   fx = 1 - pfrechet(x, location = location, shape = shape, scale = scale)
36   return(fx)
37 }
38
39
40 ### 위험함수
41 hfrechet = function (x, location = 1, shape = 1, scale = 1)
42 {
43   fx = dfrechet(x, location = location, shape = shape, scale = scale) / sfrechet(x, location = location, shape = shape, scale =
44     scale)
45   return(fx)
46 }
47
48
49
50 ###### Plot
51 plot.frechet_seq = function(x, location = 1, shape = 1, scale = 1, xlim=c(0, 10), ylim=c(0, 5), func="dfrechet")
52 {
53   color=colorPalette(300)
54 }
```

```

56 len_location = length(location) # location 파라메터의 길이
57 len_shape = length(shape) # shape 파라메터의 길이
58 len_scale = length(scale) # scale 파라메터의 길이
59
60 color_counter = 1
61 for (i in 1:len_location) ### 파라메터: location
62 {
63   if (func=="dfrechet") # 수명분포
64   {
65     for (j in 1:len_shape) ### 파라메터: shape
66     {
67       color_counter_init = color_counter
68       legend_name = NULL;
69       plot(x, dfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
70         =2, type = 'n', main="Life Distribution Function")
71       for (k in 1:len_scale) ### 파라메터: scale
72       {
73         lines(x, dfrechet(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
74         color_counter = color_counter + 1;
75         legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
76           sep=""))
77       }
78       legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
79     }
80   } else if (func == "pfrechet") # 누적분포함수
81   {
82     for (j in 1:len_shape) ### 파라메터: shape
83     {
84       color_counter_init = color_counter
85       legend_name = NULL;
86       plot(x, pfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
87         =2, type = 'n', main="Culocationative Distribution Function")
88       for (k in 1:len_scale) ### 파라메터: scale
89       {
90         lines(x, pfrechet(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
91         color_counter = color_counter + 1;
92         legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
93           sep=""))
94       }
95       legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
96     }
97   } else if (func == "sfrechet") # 생존함수
98   {
99     for (j in 1:len_shape) ### 파라메터: shape
100    {
101      color_counter_init = color_counter
102      legend_name = NULL;
103      plot(x, sfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
104        =2, type = 'n', main="Survival Function")
105      for (k in 1:len_scale) ### 파라메터: scale
106      {
107        lines(x, sfrechet(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
108        color_counter = color_counter + 1;
109        legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
110          sep=""))
111      }
112      legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
113    }
114  } else if (func == "hfrechet") # 위험함수
115  {
116    for (j in 1:len_shape) ### 파라메터: shape
117    {
118      color_counter_init = color_counter
119      legend_name = NULL;
120      plot(x, hfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
121        =2, type = 'n', main="Hazard Function")
122      for (k in 1:len_scale) ### 파라메터: scale
123      {
124        lines(x, hfrechet(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
125        color_counter = color_counter + 1;
126        legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
127          sep=""))
128      }
129    }
130  par(mfrow = c(8, 8))
131  plot.frechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dfrechet")
132
133  par(mfrow = c(8, 8))
134  plot.frechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pfrechet")
135
136  par(mfrow = c(8, 8))
137  plot.frechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sfrechet")
138
```

```
139 par(mfrow = c(8, 8))  
140 plot.frechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hfrechet")
```

Type II 극치분포는 Frechet 분포로도 알려져 있다.

독일의 주가지수에서 극단적인 경우가 발생할 확률의 추정, 태양 양성자 초고 유속의 특성을 예측하는 데 응용되었다.

14.12.5 Type II 극치분포(Frechet 분포) with Location Parameter

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(\frac{a-t}{b}\right)^{-(c+1)} e^{-\left(\frac{a-t}{b}\right)^{-c}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{-\left(\frac{a-t}{b}\right)^{-c}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c}{b} \left(\frac{a-t}{b}\right)^{-(c+1)}$	IHR
변수		$t \leq a$	
파라메터		$a \leq 0, b > 0, c > 0$	

Table 23: Type II 극치분포(Frechet 분포) with Location Parameter 함수에 기반한 척도 함수

Code 21. Type II 극치분포(Frechet 분포) with Location Parameter에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(VGAM)
4
5
6 ##### lfrechet Distribution
7 ### parameter
8 location = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
9 shape = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
11
12 ### input varialbe
13 x = seq(0, 10, length.out = 1000)
14
15 # m : shape / c
16 # beta : scale / b
17 # mu : location / a
18 ### 수명 분포
19 dlfrechet = function(x, location = 0, shape = 1, scale = 1)
20 {
21   fx = (shape/scale) * ((location - x) / scale)^(-(shape+1)) * exp(-((location - x) / scale)^(-shape))
22 }
23
24
25 ### 누적분포함수
26 plfrechet = function(x, location = 0, shape = 1, scale = 1)
27 {
28   fx = -(slfrechet(x, alpha, beta) - 1)
29 }
30
31
32 ### 생존함수
33 slfrechet = function (x, location = 1, shape = 1, scale = 1)
34 {
35   fx = exp(-((location - x) / scale)^(-shape))
36   return(fx)
37 }
38
39
40 ### 위험함수
41 hlfrechet = function (x, location = 1, shape = 1, scale = 1)
42 {
43   fx = dlfrechet(x, location = location, shape = shape, scale = scale) / slfrechet(x, location = location, shape = shape, scale = scale)
44   return(fx)
45 }
46
47
48
49
50
51 ##### Plot
52 plot.lfrechet_seq = function(x, location = 1, shape = 1, scale = 1, xlim=c(0, 10), ylim=c(0, 5), func="dlfrechet")

```

```

53 {
54   color=colorPalette(300)
55
56   len_location = length(location) # location 파라메터의 길이
57   len_shape = length(shape) # shape 파라메터의 길이
58   len_scale = length(scale) # scale 파라메터의 길이
59
60   color_counter = 1
61   for (i in 1:len_location) ### 파라메터: location
62   {
63     if (func=="dlfrechet") # 수명분포
64     {
65       for (j in 1:len_shape) ### 파라메터: shape
66       {
67         color_counter_init = color_counter
68         legend_name = NULL;
69         plot(x, dlfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
70           =2, type = 'n', main="Life Distribution Function")
71         for (k in 1:len_scale) ### 파라메터: scale
72         {
73           lines(x, dlfrechet(x, locations=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
74           color_counter = color_counter + 1;
75           legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
76             sep=""))
77         }
78         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
79       }
80     }
81   else if (func == "plfrechet") # 누적분포함수
82   {
83     for (j in 1:len_shape) ### 파라메터: shape
84     {
85       color_counter_init = color_counter
86       legend_name = NULL;
87       plot(x, plfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
88           =2, type = 'n', main="Culocationative Distribution Function")
89       for (k in 1:len_scale) ### 파라메터: scale
90       {
91         lines(x, plfrechet(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
92         color_counter = color_counter + 1;
93         legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
94           sep=""))
95       }
96       legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
97     }
98   }
99   else if (func == "slfrechet") # 생존함수
100  {
101    for (j in 1:len_shape) ### 파라메터: shape
102    {
103      color_counter_init = color_counter
104      legend_name = NULL;
105      plot(x, slfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
106          =2, type = 'n', main="Survival Function")
107      for (k in 1:len_scale) ### 파라메터: scale
108      {
109        lines(x, slfrechet(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
110        color_counter = color_counter + 1;
111        legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
112          sep=""))
113      }
114      legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
115    }
116  }
117  else if (func == "hlfrechet") # 위험함수
118  {
119    for (j in 1:len_shape) ### 파라메터: shape
120    {
121      color_counter_init = color_counter
122      legend_name = NULL;
123      plot(x, hlfrechet(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
124          =2, type = 'n', main="Hazard Function")
125      for (k in 1:len_scale) ### 파라메터: scale
126      {
127        lines(x, hlfrechet(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
128        color_counter = color_counter + 1;
129        legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
130          sep=""))
131      }
132      legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
133    }
134  }
135}
136 par(mfrow = c(8, 8))
137 plot.lfrechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dlfrechet")
138 par(mfrow = c(8, 8))
139 plot.lfrechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="plfrechet")

```

```
136 | par(mfrow = c(8, 8))  
137 | plot.lfrechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="slfrechet")  
138 |  
139 | par(mfrow = c(8, 8))  
140 | plot.lfrechet_seq(x, location, shape, scale, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hlfrechet")
```

14.13 F Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{\Gamma(\frac{\nu_1+\nu_2}{2})}{\Gamma(\frac{\nu_1}{2})\Gamma(\frac{\nu_2}{2})} \left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}} \frac{t^{\frac{\nu_1-2}{2}}}{(1+\frac{\nu_1}{\nu_2}t)^{\frac{\nu_1+\nu_2}{2}}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	No closed form	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	
변수		$t \geq 0$	
파라메터		$\nu_1 > 0, \nu_2 > 0$	

Table 24: F 분포함수에 기반한 척도 함수

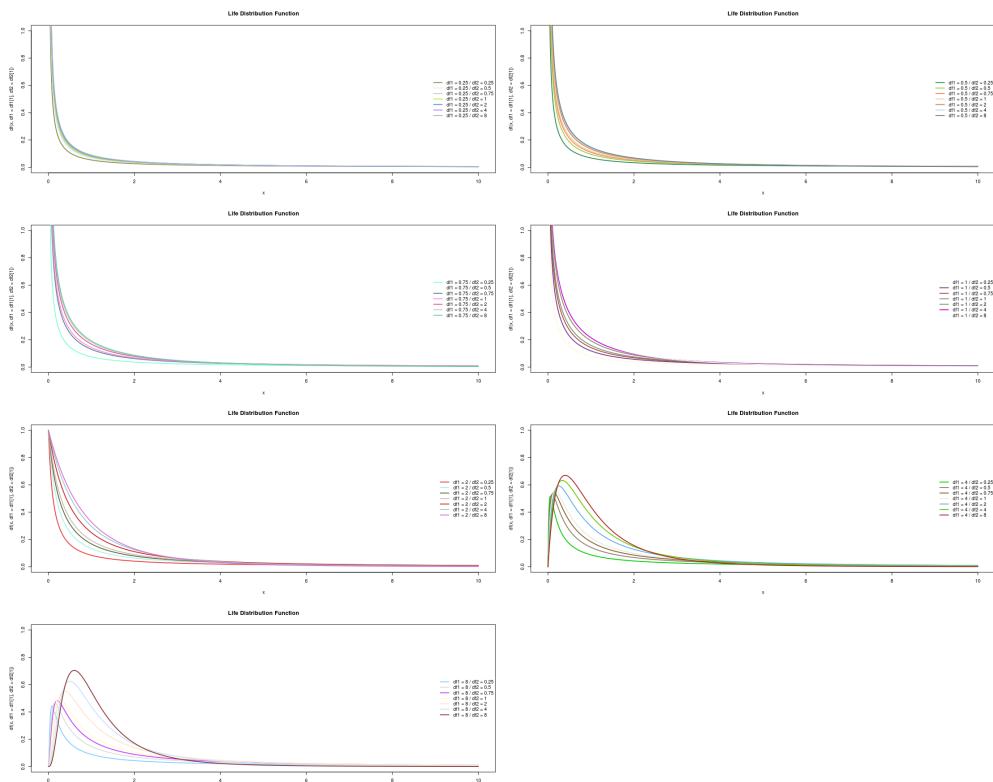


Figure 60: F Distribution에 기반한 수명분포함수

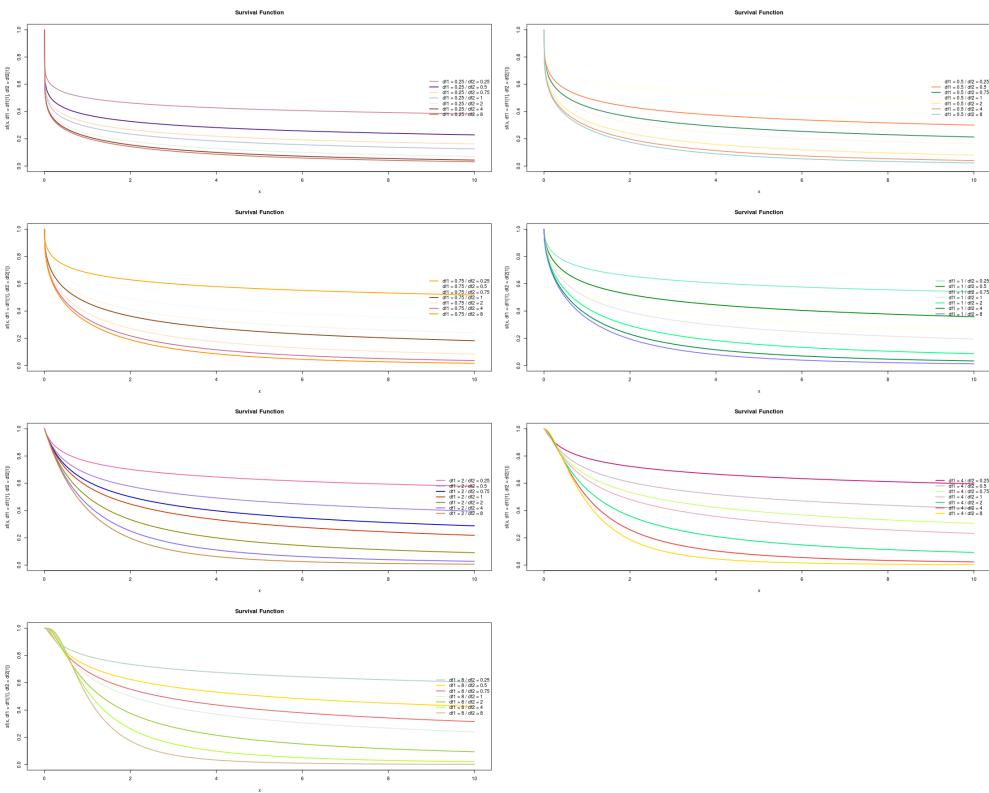


Figure 61: F Distribution에 기반한 생존함수

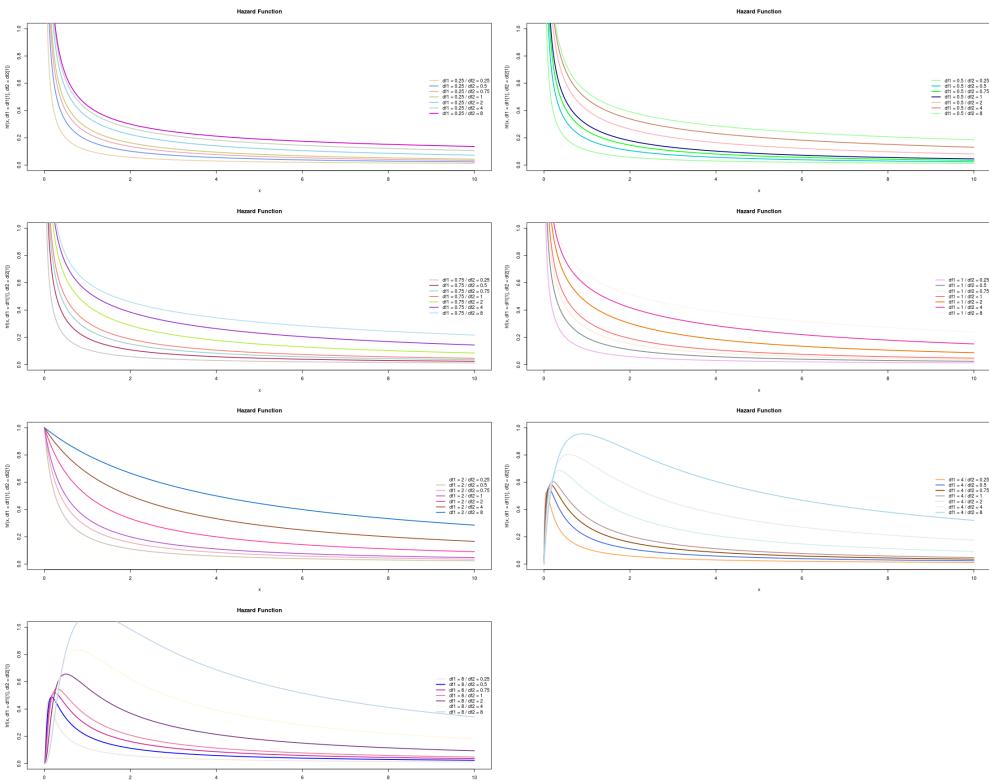


Figure 62: F Distribution에 기반한 위험함수

Code 22. F Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### F Distribution
6 ### parameter
7 df1 = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8 df2 = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 df(x, df1 = df1, df2 = df2)
16
17
18 ### 분위수 함수
19 qf(x, df1 = df1, df2 = df2)
20
21
22 ### 난수 함수
23 rf(x, df1 = df1, df2 = df2)
24
25
26 ### 누적분포함수
27 pf(x, df1 = df1, df2 = df2)
28
29
30 ### 생존함수
31 sf = function (x, df1 = 1, df2 = 1)
32 {
33   fx = 1 - pf(x, df1 = df1, df2 = df2)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hf = function (x, df1 = 1, df2 = 1)
40 {
41   fx = df(x, df1 = df1, df2 = df2) / sf(x, df1 = df1, df2 = df2)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.f_seq = function(x, df1 = 1, df2 = 0, xlim=c(0, 10), ylim=c(0, 5), func="df")
51 {
52   color=colorPalette(300)
53
54   len_df1 = length(df1) # df1 파라메터의 길이
55   len_df2 = length(df2) # df2 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_df1) ### 파라메터: df1
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="df") # 수명분포
64     {
65       plot(x, df(x, df1=df1[i], df2=df2[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution
66           Function")
67       for (j in 1:len_df2) ### 파라메터: df2
68       {
69         lines(x, df(x, df1=df1[i], df2=df2[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("df1 = ", df1[i], " / df2 = ", df2[j], sep=""))
72       }
73     }
74     else if (func == "pf") # 누적분포함수
75     {
76       plot(x, pf(x, df1=df1[i], df2=df2[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
77           Distribution Function")
78       for (j in 1:len_df2) ### 파라메터: df2
79       {
80         lines(x, pf(x, df1=df1[i], df2=df2[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("df1 = ", df1[i], " / df2 = ", df2[j], sep=""))
83     }
84     else if (func == "sf") # 생존함수
85     {
86       plot(x, sf(x, df1=df1[i], df2=df2[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
87       for (j in 1:len_df2) ### 파라메터: df2
88     }
89   }
90 }
```

```

87     {
88         lines(x, sf(x, df1=df1[i], df2=df2[j]), col=color[color_counter], lwd=2);
89         color_counter = color_counter + 1;
90         legend_name = c(legend_name, paste("df1 = ", df1[i], " / df2 = ", df2[j], sep=""))
91     }
92 }
93 else if (func == "hf") # 위험함수
94 {
95     plot(x, hf(x, df1=df1[1], df2=df2[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
96     for (j in 1:len_df2) ### 파라메트릭 df2
97     {
98         lines(x, hf(x, df1=df1[i], df2=df2[j]), col=color[color_counter], lwd=2);
99         color_counter = color_counter + 1;
100        legend_name = c(legend_name, paste("df1 = ", df1[i], " / df2 = ", df2[j], sep=""))
101    }
102 }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(4, 2))
108 plot.f_seq(x, df1, df2, xlim=c(min(x), max(x)), ylim=c(0, 1), func="df")
109
110 par(mfrow = c(4, 2))
111 plot.f_seq(x, df1, df2, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pf")
112
113 par(mfrow = c(4, 2))
114 plot.f_seq(x, df1, df2, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sf")
115
116 par(mfrow = c(4, 2))
117 plot.f_seq(x, df1, df2, xlim=c(min(x), max(x)), ylim=c(0, 1), func="hf")
118

```

14.14 Gamma Distribution(감마 분포)

감마분포는 지수분포를 보다 일반화시킨 수명분포로, 모수가 특정한 값이 될 때 지수분포, 카이제곱분포, Erlang 분포 등으로 변환되는 특징이 있다.

감마분포는 위험률(고장률) 형태가 복잡하여 와이블분포보다 덜 사용되고 있으나, 여러 가지 수명 분포에 잘 적용되는 모형이라는 것이 입증되어, 자료를 분석할 때 유용한 분포이다.

14.14.1 Gamma Distribution with 2 Parameters(2-모수 감마 분포)

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{\theta \Gamma(\beta)} \left(\frac{t}{\theta}\right)^{\beta-1} e^{-\frac{t}{\theta}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\int_t^{\infty} \frac{1}{\theta \Gamma(\beta)} \left(\frac{u}{\theta}\right)^{\beta-1} e^{-\frac{u}{\theta}} du$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{\frac{1}{\theta \Gamma(\beta)} \left(\frac{t}{\theta}\right)^{\beta-1} e^{-\frac{t}{\theta}}}{\int_t^{\infty} \frac{1}{\theta \Gamma(\beta)} \left(\frac{u}{\theta}\right)^{\beta-1} e^{-\frac{u}{\theta}} du}$	
변수		$t \geq 0$	
파라메터	$\beta > 0$ (형상 모수; shape parameter), $\theta > 0$ (척도 모수; scale parameter)		

Table 25: 2모수 감마 분포함수에 기반한 척도 함수

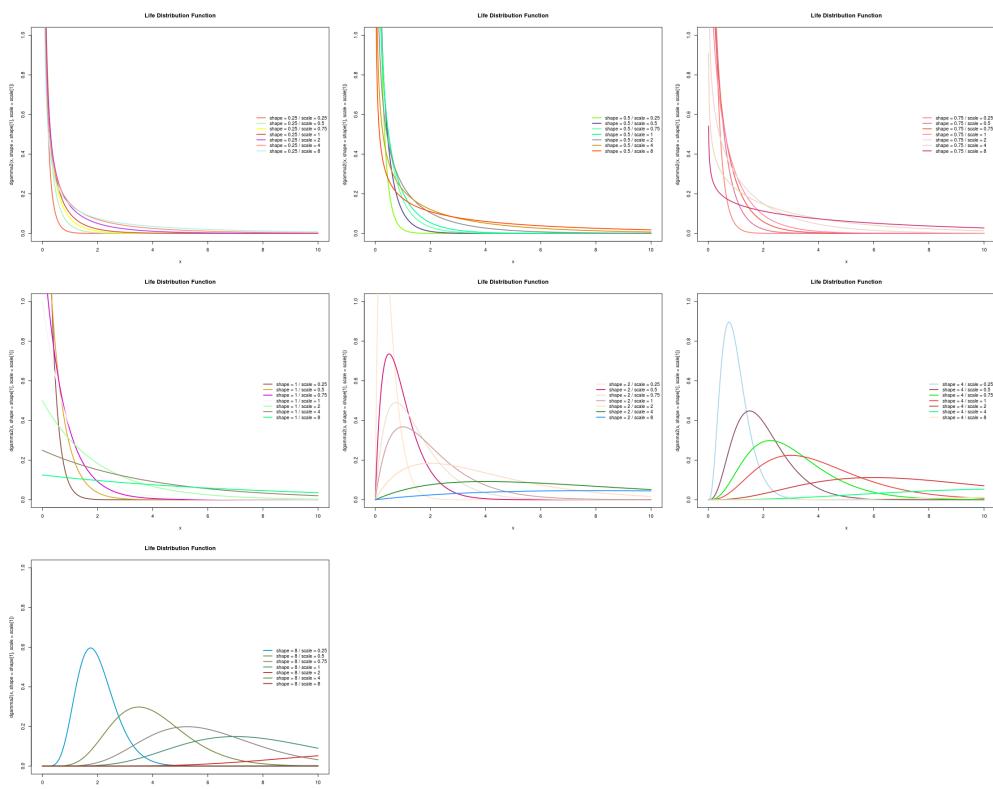


Figure 63: 2모수 감마 Distribution에 기반한 수명분포함수

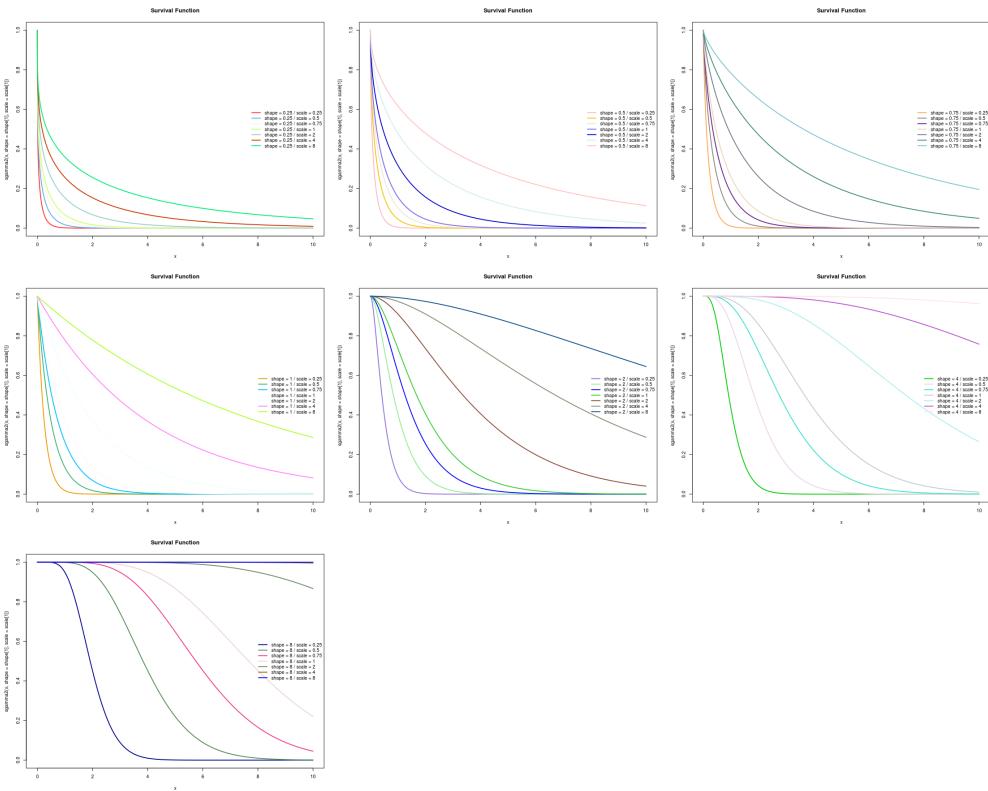


Figure 64: 2모수 감마 Distribution에 기반한 생존함수

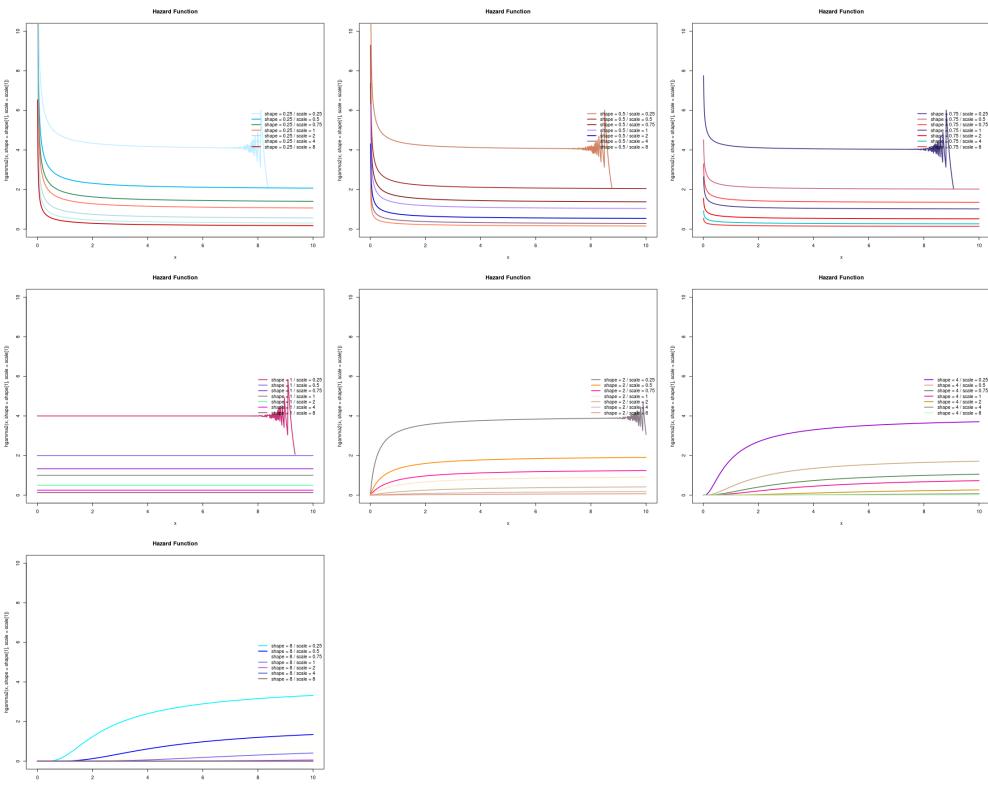


Figure 65: 2모수 감마 Distribution에 기반한 위험함수

Code 23. 2모수 감마 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Gamma Distribution
6 ### parameter
7 shape = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dgamma2 = function(x, shape=shape, scale=scale)
16 {
17   fx = dgamma(x, shape=shape, scale=scale)
18   return(fx)
19 }
20
21
22 ### 분위수 함수
23 qgamma2 = function(x, shape=shape, scale=scale)
24 {
25   fx = qgamma(x, shape=shape, scale=scale)
26   return(fx)
27 }
28
29
30 ### 난수 함수
31 rgamma2 = function(x, shape=shape, scale=scale)
32 {
33   fx = rgamma(x, shape=shape, scale=scale)
34   return(fx)
35 }
36
37
38 ### 누적분포함수
39 pgamma2 = function(x, shape=shape, scale=scale)
40 {
41   fx = pgamma(x, shape=shape, scale=scale)
42   return(fx)
43 }
44
45
46 ### 생존함수
47 sgamma2 = function(x, shape=1, scale=1)
48 {
49   fx = 1 - pgamma2(x, shape=shape, scale=scale)
50   return(fx)
51 }
52
53
54 ### 위험함수
55 hgamma2 = function (x, shape=shape, scale=scale)
56 {
57   fx = dgamma2(x, shape=shape, scale=scale) / sgamma2(x, shape=shape, scale=scale)
58   return(fx)
59 }
60
61
62
63
64
65 ##### Plot
66 plot.gamma2_seq = function(x, shape = 1, scale = 1, xlim=c(0, 10), ylim=c(0, 5), func="dgamma2")
67 {
68   color=colorPalette(300)
69
70   len_shape = length(shape) # shape 파라미터의 길이
71   len_scale = length(scale) # scale 파라미터의 길이
72
73   color_counter = 1
74   for (i in 1:len_shape) ### 파라미터: shape
75   {
76     color_counter_init = color_counter
77     legend_name = NULL;
78
79     if (func=="dgamma2") # 수명분포
80     {
81       plot(x, dgamma2(x, shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
82           Distribution Function")
83       for (j in 1:len_scale) ### 파라미터: scale
84       {
85         lines(x, dgamma2(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
86         color_counter = color_counter + 1;
87         legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
88       }
89     }
90   }
91 }
```

```

88 }
89 else if (func == "pgamma2") # 누적분포함수
90 {
91   plot(x, pgamma2(x, shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
92     Cumulative Distribution Function")
93   for (j in 1:len_scale) ### 파라미터: scale
94   {
95     lines(x, pgamma2(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
96     color_counter = color_counter + 1;
97     legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
98   }
99 else if (func == "sgamma2") # 생존함수
100 {
101   plot(x, sgamma2(x, shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
102     Survival Function")
103   for (j in 1:len_scale) ### 파라미터: scale
104   {
105     lines(x, sgamma2(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
106     color_counter = color_counter + 1;
107     legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
108   }
109 else if (func == "hgamma2") # 위험함수
110 {
111   plot(x, hgamma2(x, shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
112     Function")
113   for (j in 1:len_scale) ### 파라미터: scale
114   {
115     lines(x, hgamma2(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
116     color_counter = color_counter + 1;
117     legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
118   }
119   legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
120 }
121 }
122
123 par(mfrow = c(3, 3))
124 plot.gamma2_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="dgamma2")
125
126 par(mfrow = c(3, 3))
127 plot.gamma2_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pgamma2")
128
129 par(mfrow = c(3, 3))
130 plot.gamma2_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sgamma2")
131
132 par(mfrow = c(3, 3))
133 plot.gamma2_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hgamma2")

```

감마분포는 형상 모수 β 의 값에 따라서, 위험률 함수(고장률 함수)에 의한 비모수적 수명 분포 군(class of life distribution)이 달라진다. 즉, β 의 값이 위험률 함수(고장률 함수) $h(t)$ 의 단조성(monotonicity)에 영향을 미친다.

- $0 < \beta < 1$: $t = 0$ 에서 $h(t)$ 는 ∞ 의 값을 취하며, 그 이후 t 가 증가함에 따라 $h(t)$ 는 단조감소함수이며, $\lim_{t \rightarrow \infty} h(t) = \frac{1}{\theta}$ 을 만족하는 convex 함수이다.
따라서, 이 경우의 감마분포는 DFR 군에 속하게 되며, 초기 고장현상을 보이고, 시간이 경과함에 따라 위험률(고장률)이 감소하는 수명을 나타내는 데 적합해진다.
 - $\beta = 1$: $h(t) = \frac{1}{\theta}$ 로서, 나이에 종속되지 않는 상수 위험률(고장률)을 가지는 CFR 군에 속하게 된다. 이 때는 우연한 고장이 발생하는 경우의 수명을 모형화하는 데 적합하다.
 - $\beta > 1$: $t = 0$ 에서 $h(t) = 0$ 이 되며, 그 이후 t 가 증가함에 따라 $h(t)$ 는 단조증가함수로서, IFR 군에 속하며, $\lim_{t \rightarrow \infty} h(t) = \frac{1}{\theta}$
-

Note 15. 감마 함수

감마함수 $\Gamma(a)$ 는 다음과 같이 정의된다.

$$\Gamma(a) = \int_0^\infty x^{a-1} e^{-x} dx$$

$\Gamma(a)$ 는 위 식의 우변 항에 있는 적분을 나타내는 기호이다.

감마 함수의 성질은 다음과 같다.

- 부분적분에 의하여, 다음과 같은 성질이 있다.

$$\Gamma(a+1) = a\Gamma(a)$$

- 만일, $a = n$, 즉 a 가 정수이면, 다음과 같은 식이 성립한다.

- $\Gamma(n+1) = n!$
- $\Gamma(\frac{1}{2}) = 2\Gamma(\frac{3}{2}) = \sqrt{\pi}$
- $\Gamma(n + \frac{1}{2}) = \frac{1 \cdot 3 \cdot 5 \cdots (2n-1)}{2^n} \sqrt{\pi}$

Note 16. 감마분포에 기반한 생존함수(신뢰도함수)

감마분포에 기반한 생존함수는, 감마함수를 적분할 수 없기 때문에 적분을 포함한 식으로 표현된다.

$$S(t) = \int_t^\infty \frac{1}{\theta\Gamma(\beta)} \left(\frac{u}{\theta}\right)^{\beta-1} e^{-\frac{u}{\theta}} du$$

만일 β 가 정수이면, 생존함수(신뢰도함수)에 대해 부분적분을 반복하여 다음과 같은 식을 얻을 수 있다.

$$S(t) = \sum_{k=0}^{\beta-1} \frac{1}{k!} \left(\frac{t}{\theta}\right)^k e^{-\frac{t}{\theta}} \quad (3)$$

식 (3)의 오른쪽 항은, 평균이 $\frac{t}{\theta}$ 인 포아송 분포에서, $\beta - 1$ 까지의 누적분포함수와 같다.

Note 17. 감마분포에 기반한 위험률함수(고장률함수)

$$h(t) = \frac{f(t)}{S(t)} = \frac{\frac{1}{\theta\Gamma(\beta)} \left(\frac{t}{\theta}\right)^{\beta-1} e^{-\frac{t}{\theta}}}{\int_t^\infty \frac{1}{\theta\Gamma(\beta)} \left(\frac{u}{\theta}\right)^{\beta-1} e^{-\frac{u}{\theta}} du}$$

감마분포의 위험률(고장률)은, β 에 따라 다음과 같은 성질이 있다.

- $\beta > 1$ 일 때: 위험률(고장률)이 0에서 출발하여 단조증가하여, $t \rightarrow \infty$ 이면 $\frac{1}{\theta}$ 로 수렴한다.
 - $0 < \beta < 1$ 일 때: 위험률(고장률)이 ∞ 에서 출발하여 단조감소하여, $t \rightarrow \infty$ 이면 $\frac{1}{\theta}$ 로 수렴한다.
 - $\beta = 1$ 일 때: 위험률(고장률)이 $\frac{1}{\theta}$ 로 일정한 값을 가진다.
-

Note 18. 감마분포에 기반한 k차 적률

$$\begin{aligned} E[T^k] &= \int_0^\infty t^k f(t) dt \\ &= \int_0^\infty t^k \left(\frac{t}{\theta}\right)^{\beta-1} e^{-\frac{t}{\theta}} dt \\ &= \frac{\theta^k \Gamma(k+\beta)}{\Gamma(\beta)} \int_0^\infty \frac{1}{\theta\Gamma(k+\beta)} \left(\frac{t}{\theta}\right)^{k+\beta-1} e^{-\frac{t}{\theta}} dt \\ &= \frac{\theta^k \Gamma(k+\beta)}{\Gamma(\beta)} \end{aligned}$$

감마분포는 다음과 같은 성질이 있다.

- Figure ??에서 알 수 있듯이, 감마분포는 오른쪽으로 치우쳐 있다.

– $\beta < 1$ 일 때: t 값이 0에 가까워질수록 확률밀도함수의 값은 무한대가 된다.

– $\beta = 1$ 일 때: 감마분포는 지수분포와 동일하게 $t = 0$ 에서 확률밀도함수의 값은 $\frac{1}{\theta}$ 이며, t 가 커짐에 따라 감소한다.

– $\beta > 1$ 일 때: 감마분포는 비대칭 종모양을 하는데, $t = 0$ 일 때 $f(t) = 0$ 이며, 증가하다가 감소하는 형태를 보여준다.

이러한 특징은 와이블 분포와 유사하다.

- 감마분포를 따르는 제품의 평균수명은 $\theta\beta$ 표준편차는 $\theta\beta^{\frac{1}{2}}$ 이다.
- 감마분포의 중앙값(t_m)은 다음과 같다.

$$0.5 = \int_0^{t_m} \frac{1}{\theta\Gamma(\beta)} \left(\frac{t}{\theta}\right)^{\beta-1} e^{-\frac{t}{\theta}} dt$$

주어진 θ 와 β 값에 대해 불완전 감마함수 표를 이용하여 중앙값을 구할 수 있으나, $t_m \approx \theta(\beta - 0.3)$ 식을 통해 근사값을 구하기도 한다.

- 감마분포에서 $\beta = 1$ 인 특별한 경우에, 감마분포는 지수분포가 된다.

반대로, 만일 X_1, X_2, \dots, X_r 이 평균이 $\frac{1}{\lambda}$ 인 지수분포를 따르는 서로 독립적인 확률변수라면,

이 확률변수들의 합인 $\sum_{i=1}^r X_i$ 은 형상모수(shape parameter) β 가 r 인 감마분포를 따른다.

- 위험률(고장률)이 λ (상수)인 지수분포를 따르는 서로 독립인 β (정수) 개의 확률변수 T_1, T_2, \dots, T_β 들에 대하여

$T = \sum_{i=1}^\beta T_i \sim Gamma(\beta, \theta = \frac{1}{\lambda})$ 가 된다.²⁴

- 감마분포에서 $\beta = \frac{\nu}{2}$, $\theta = 2$ 인 감마분포는, 자유도가 ν 인 카이제곱분포가 된다.

- β 가 정수로 제한되는 경우, 감마분포는 Erlang분포가 된다.²⁵

- 감마분포의 확률밀도함수는, β 가 충분히 커지면, 평균이 $\theta\beta$ 이고 표준편차가 $\theta\beta^{\frac{1}{2}}$ 인 정규분포에 근접한다.

²⁴이는, 적률생성함수를 이용하여 증명할 수 있다.

²⁵Erlang 분포는 대기 이론이나 traffic 분석에서 자주 이용된다. Erlang 분포의 확률밀도함수는 다음과 같다.

$$f(t; \beta, \theta) = \frac{1}{\theta(\beta-1)!} \left(\frac{t}{\theta}\right)^{\beta-1} e^{-\frac{t}{\theta}}$$

Note 19. 감마분포를 따르는 완전자료에 대한 모수 추정

T_1, T_2, \dots, T_n 을 시험 중인 n 개의 부품(환자)에 대한 고장(재발) 시간을 나타내는 확률변수라 하고, t_1, t_2, \dots, t_n 을 이들의 실제 관측값이라고 하자. T_1, T_2, \dots, T_n 은 감마 분포를 따른다고 하자. 완전 자료를 이용하면, 감마분포의 형상모수 β 와 척도모수 θ 의 추정량은 다음과 같이 계산된다.

- 적률 추정량(moment estimator)

적률 함수를 이용하여 감마분포의 모수를 추정하는 방법은 다음과 같다.

$m_1 = E(X) = \theta\beta$ 이고, $m_2 = E(X^2) = \theta^2\beta(\beta + 1)$ 이므로, 표본적률값을 이용하여 다음과 같은 방정식을 세울 수 있다.

$$\theta\beta = \frac{1}{n} \sum_{i=1}^n t_i = \bar{t}$$

$$\theta^2\beta(\beta + 1) = \sum_{i=1}^n \frac{t_i^2}{n}$$

위의 방정식을 풀면, θ 와 β 에 대해 다음과 같은 적률추정량을 구할 수 있다.

$$\hat{\theta} = \frac{\frac{\bar{t}^2}{n} - \bar{t}^2}{\bar{t}}$$

$$\hat{\beta} = \frac{\bar{t}}{\frac{\bar{t}^2}{n} - \bar{t}^2}$$

감마분포의 경우, 표본의 크기가 크지 않은 경우, 적률추정량은 최대가능도추정량에 비하여 유효성에 차이가 없으므로, 최대가능도추정량 대신 사용할 수 있다.s

- 최대 가능도 추정량(Maximum Likelihood Estimator)

감마함수의 가능도함수는 다음과 같다.

$$\begin{aligned} L(t_i; \beta, \theta) &= \prod_{i=1}^n \frac{1}{\theta \Gamma(\beta)} \left(\frac{t_i}{\theta} \right)^{\beta-1} e^{-\frac{t_i}{\theta}} \\ &= \left(\frac{1}{\theta^\beta \Gamma(\beta)} \right)^n \prod_{i=1}^n t_i^{\beta-1} e^{-\frac{\sum_{i=1}^n t_i}{\theta}} \end{aligned}$$

양변에 자연로그를 취하면,

$$\ln L(t_i; \beta, \theta) = -n\beta \ln \theta - n \ln \Gamma(\beta) + (\beta - 1) \ln \prod_{i=1}^n t_i - \left(\frac{n\bar{t}}{\theta} \right)$$

위 식을 θ 와 β 에 대한 1차도함수를 구하고 0으로 놓으면 얻어지는 가능도 방정식을 풀면, 다음과 같은 식을 얻고, 이 식으로부터 최대가능도추정량(MLE)을 구할 수 있다.

$$\ln \beta - \psi(\beta) = \ln \bar{t} - \frac{1}{b} \ln \prod_{i=1}^n t_i \quad (4)$$

$$\theta = \frac{\bar{t}}{\beta}$$

$$\text{여기서, } \psi(\beta) = \frac{\Gamma''(\beta)}{\Gamma(\beta)}$$

실제 데이터를 이용하여 $d = \ln \bar{t} - \frac{1}{n} \ln \prod_{i=1}^n t_i$ 가 주어졌을 때, 식 (4)을 만족하는 $\hat{\beta}$ 를 구하는 방법들이 제안되었는데, 그 중 2가지만 소개하면 다음과 같다.

- Greenwood and Duran(1960)[28]

$$\hat{\beta} = \begin{cases} \frac{0.5000876 + 0.1648852d - 0.0544274d^2}{d} & 0 < d \leq 0.5772 \\ \frac{8.898919 + 9.059950d + 0.9775373d^2}{d(17.79728 + 11.968477d + d^2)} & 0.5772 < d \leq 17 \end{cases}$$

이 식은 무언가 rough하게 정해진 듯 하면서도, 추정오차가 0.1% 미만으로 매우 정확한 것으로 알려져 있다.

- Bowman and Shenton(1983)[18]

$$\hat{\beta} = \begin{cases} \frac{1}{d + \ln d} & n \circ \text{작을 때} \\ \frac{1}{2d} + \frac{1}{6} - \frac{d}{18} - \frac{4d^2}{135} + \frac{47d^2}{810} & n \circ \text{클 때} \end{cases}$$

- 신뢰구간

감마분포의 모수에 대한 신뢰구간은 Bain(1978)이 제안한 방법과, 표본의 크기가 큰 경우 근사적으로 구하는 방법이 있다.

- Bain(1978)의 방법[9]

표본의 크기가 작아도 $2n\beta d$ 는 근사적으로 자유도가 $(n - 1)$ 인 카이제곱분포를 따른다. 따라서, 만일 $\beta \geq 2$ 이고, $n \geq 10$ 이라면, 다음과 같은 신뢰구간의 상한과 하한을 구할 수 있다.

$$\left[\frac{1}{2nd} \chi^2 \left(n - 1; \frac{1 - \alpha}{2} \right) < \beta < \frac{1}{2nd} \chi^2 \left(n - 1; \frac{\alpha}{2} \right) \right]$$

$$\left[\frac{2n\bar{t}}{\chi^2 \left(2n [\eta_U]; \frac{\alpha}{4} \right)} < \theta < \frac{2n\bar{t}}{\chi^2 \left(2n [\eta_L]; \frac{1-\alpha}{4} \right)} \right]$$

여기서, $[\eta_U], [\eta_L]$ 은 β 의 신뢰상한과 하한의 정수부분을 의미함

- 근사적 신뢰구간

표본의 크기가 충분히 큰 경우, $\hat{\theta}$ 와 $\hat{\beta}$ 의 근사적 분포는 다음과 같다.

$$\hat{\theta} \sim AN \left[\theta, \frac{\theta^2}{n} \left(\frac{\psi'(\beta)}{\beta\psi'(\beta) - 1} \right) \right]$$

$$\hat{\beta} \sim AN \left[\beta, \frac{1}{n} \left(\frac{\beta}{\beta\psi'(\beta) - 1} \right) \right]$$

여기서, $\psi(\beta) = \frac{d \ln \Gamma(\beta)}{d\beta}$: Digamma 함수

이에 따른 근사적 신뢰구간은 다음과 같다.

$$\left[\hat{\beta} - z_{\frac{\alpha}{2}} \frac{1}{\sqrt{n}} \left(\frac{\hat{\beta}}{\hat{\beta}\psi'(\hat{\beta}) - 1} \right)^{\frac{1}{2}} < \beta < \hat{\beta} + z_{\frac{\alpha}{2}} \frac{1}{\sqrt{n}} \left(\frac{\hat{\beta}}{\hat{\beta}\psi'(\hat{\beta}) - 1} \right)^{\frac{1}{2}} \right]$$

$$\left[\hat{\theta} - z_{\frac{\alpha}{2}} \frac{\hat{\theta}}{\sqrt{n}} \left(\frac{\psi'(\hat{\beta})}{\hat{\beta}\psi'(\hat{\beta}) - 1} \right)^{\frac{1}{2}} < \theta < \hat{\theta} + z_{\frac{\alpha}{2}} \frac{\hat{\theta}}{\sqrt{n}} \left(\frac{\psi'(\hat{\beta})}{\hat{\beta}\psi'(\hat{\beta}) - 1} \right)^{\frac{1}{2}} \right]$$

Note 20. Digamma 함수

$$\text{Digamma 함수 : } \psi(\beta) = \frac{d \ln \Gamma(\beta)}{d\beta}$$

$\psi(\beta)$ 는 다음과 같은 성질을 가지고 있다.

- $\beta > 0$ 에 대해, $\psi(\beta + 1) = \psi(\beta) + \frac{1}{\beta} = -\gamma + \sum_{n=1}^{\infty} \frac{\beta}{n(n+\beta)}$
- $\beta > 0$ 에 대해, $\psi'(\beta) = \sum_{k=0}^{\infty} (\beta + k)^{-2}$
- $n \geq 2, 0 < z < 1$ 에 대해, $\psi'(n + z) = \psi(1 + z) - \sum_{j=1}^{n-1} (j + z)^{-2}$

Note 21. 감마분포를 따르는 Type II 우중도절단자료(정수중단자료)에 대한 모수 추정

2모수 감마분포를 따르는 n 개의 부품(환자)을 가지고 수명시험을 수행하여 $r(\leq n)$ 개의 고장(재발)이 발생한 시점에서 시험을 중단하였을 때, 일어진 고장(재발)자료를 크기 순서대로 나열하여 $t_{(1)} \leq t_{(2)} \leq \dots \leq t_{(r)}$ 으로 나타낸다. 나머지 $(n - r)$ 개의 부품(환자)에 대한 고장(재발)자료는 시점 $t_{(r)}$ 에서 절단된다.

Type II 우중도절단자료(정수중단자료)가 주어진 경우의 β 와 θ 에 대한 가능도함수는 다음과 같다.

$$\begin{aligned} L(\beta, \theta) &= \prod_{i=1}^r f(t_{(i)}; \beta, \theta) (P(T > t_{(r)}))^{n-r} \\ &= \prod_{i=1}^r \frac{1}{\theta \Gamma(\beta)} \left(\frac{t_{(i)}}{\theta}\right)^{\beta-1} e^{-\frac{t_{(i)}}{\theta}} \left(1 - IG\left(\beta, \frac{t_{(r)}}{\theta}\right)\right)^{n-r} \\ &= \left(\frac{1}{\Gamma(\beta)}\right)^r \left(\frac{1}{\theta}\right)^{r\beta} \prod_{i=1}^r t_{(i)}^{\beta-1} e^{-\frac{\sum_{i=1}^r t_{(i)}}{\theta}} \left(1 - IG\left(\beta, \frac{t_{(r)}}{\theta}\right)\right)^{n-r} \end{aligned}$$

여기서, $IG(a, b) = \frac{1}{\Gamma(a)} \int_0^b u^{a-1} e^{-u} du$: 불완전 감마 함수

양변에 자연로그를 취하면,

$$\ln L(\beta, \theta) = -r\beta \ln \theta - r \ln \Gamma(\beta) + (\beta - 1) \ln \tilde{t} - \frac{r\bar{t}}{\theta} + (n - r) \ln \left[1 - IG\left(\beta, \frac{t_{(r)}}{\theta}\right)\right] \quad (5)$$

$$\text{여기서, } \tilde{t} = \prod_{i=1}^r t_{(i)} \quad \bar{t} = \frac{\prod_{i=1}^r t_{(i)}}{r}$$

β 와 θ 의 최대가능도추정량을 구하기 위해, 식(5)의 1차도함수를 구하여 각각 0으로 놓으면, 다음과 같은 2개의 방정식을 얻게 된다.

$$\frac{d \ln L(\beta, \theta)}{d\beta} = -r \ln \theta - r\psi(\beta) + \ln \tilde{t} - \frac{n - r}{1 - IG\left(\beta, \frac{t_{(r)}}{\theta}\right)} \cdot \frac{dIG\left(\beta, \frac{t_{(r)}}{\theta}\right)}{d\beta} \quad (6)$$

$$\frac{d \ln L(\beta, \theta)}{d\theta} = -\frac{r\beta}{\theta} + \frac{r\bar{\beta}}{\theta^2} - \frac{n - r}{1 - IG\left(\beta, \frac{t_{(r)}}{\theta}\right)} \cdot \frac{dIG\left(\beta, \frac{t_{(r)}}{\theta}\right)}{d\theta} \quad (7)$$

$$\begin{aligned} \text{여기서, } \psi(\beta) &= \frac{\Gamma'(\beta)}{\Gamma(\beta)} \\ IG\left(\beta, \frac{t_{(r)}}{\theta}\right) &= \frac{1}{\Gamma(\beta)} \int_0^{\frac{t_{(r)}}{\theta}} u^{\beta-1} e^{-u} du \\ \frac{dIG\left(\beta, \frac{t_{(r)}}{\theta}\right)}{d\beta} &= \frac{1}{\Gamma(\beta)} \int_0^{\frac{t_{(r)}}{\theta}} u^{\beta-1} (\ln u) e^{-u} du - \psi(\beta)IG\left(\beta, \frac{t_{(r)}}{\theta}\right) \\ \frac{dIG\left(\beta, \frac{t_{(r)}}{\theta}\right)}{d\theta} &= \frac{-1}{\theta \Gamma(\beta)} \left(\frac{t_{(r)}}{\theta}\right)^\beta e^{-\frac{t_{(r)}}{\theta}} \end{aligned}$$

감마분포의 경우, 최대가능도추정량은 가능도방정식을 풀어서 해를 구하는 것이 복잡하므로,

- 보통 로그가능도함수 식 (5)을 최대화하는 β , θ 를 직접 찾는 방법을 사용하거나,
 - '가능한 β ' 값의 범위에서 여러 개의 β 값을 선택하고,
주어진 β 에 대해서 식 (6), (7)을 풀어 $\hat{\theta}(\beta)$ 를 구한 후, 이를 대수가능도함수인 식 (5)에 대입하고,
이 식 (5)를 최대로 하는 β 를 구하여 최대가능도추정량 $\hat{\beta}$ 으로 하고, θ 의 최대가능도추정량 $\hat{\theta}$ 는 $\hat{\theta}(\hat{\beta})$ 로 구하는 방법을 이용한다.
-

14.14.2 Gamma Distribution with Location Parameter

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$(t - a)^{c-1} e^{-\frac{t-a}{b}} [b^c \Gamma(c)]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\frac{\gamma[c, \frac{t-a}{b}]}{\Gamma(c)}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$(t - a)^{c-1} e^{-\frac{t-a}{b}} [b^c \Gamma(c)]^{-1}$	
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 26: 3모수 감마 분포함수에 기반한 척도 함수

Code 24. 3모수 감마분포에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Gamma Distribution with Location Parameters
6 ### parameter
7 location = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 shape = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dgamma3 = function(x, shape=shape, scale=scale, location=location)
17 {
18   fx = dgamma(x-location, shape=shape, scale=scale)
19   return(fx)
20 }
21
22
23 ### 분위수 함수
24 qgamma3 = function(x, shape=shape, scale=scale, location=location)
25 {
26   fx = qgamma(x-location, shape=shape, scale=scale)
27   return(fx)
28 }
29
30
31 ### 난수 함수
32 rgamma3 = function(x, shape=shape, scale=scale, location=location)
33 {
34   fx = rgamma(x-location, shape=shape, scale=scale)
35   return(fx)
36 }
37
38
39 ### 누적분포함수
40 pgamma3 = function(x, shape=shape, scale=scale, location=location)
41 {
42   fx = pgamma(x-location, shape=shape, scale=scale)
43   return(fx)
44 }
45
46
47 ### 생존함수
48 sgamma3 = function(x, shape=1, scale=1, location=0)
49 {
50   fx = 1 - pgamma3(x, shape=shape, scale=scale, location=location)
51   return(fx)
52 }
53
54
55 ### 위험함수

```

```

56 hgamma3 = function (x, shape=shape, scale=scale, location=0)
57 {
58   fx = dgamma3(x, shape=shape, scale=scale, location=location) / sgamma3(x, shape=shape, scale=scale, location=location)
59   return(fx)
60 }
61
62
63
64
65 ###### Plot
66 plot.gamma3_seq = function(x, shape = 1, scale = 1, location = 1, xlim=c(0, 10), ylim=c(0, 5), func="dgamma3")
67 {
68   color=colorPalette(300)
69
70   len_location = length(location) # location 파라미터의 길이
71   len_shape = length(shape) # shape 파라미터의 길이
72   len_scale = length(scale) # scale 파라미터의 길이
73
74   color_counter = 1
75   for (i in 1:len_location) ### 파라미터: location
76   {
77     if (func=="dgamma3") # 수명분포
78     {
79       for (j in 1:len_shape) ### 파라미터: shape
80       {
81         color_counter_init = color_counter
82         legend_name = NULL;
83         plot(x, dgamma3(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
84           =2, type = 'n', main="Life Distribution Function")
85         for (k in 1:len_scale) ### 파라미터: scale
86         {
87           lines(x, dgamma3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
88           color_counter = color_counter + 1;
89           legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
90             sep=""))
91         }
92         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
93       }
94     } else if (func == "pgamma3") # 누적분포함수
95     {
96       for (j in 1:len_shape) ### 파라미터: shape
97       {
98         color_counter_init = color_counter
99         legend_name = NULL;
100        plot(x, pgamma3(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
101          =2, type = 'n', main="Cumulative Distribution Function")
102        for (k in 1:len_scale) ### 파라미터: scale
103        {
104          lines(x, pgamma3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
105          color_counter = color_counter + 1;
106          legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
107            sep=""))
108        }
109        legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
110      }
111    } else if (func == "sgamma3") # 생존함수
112    {
113      for (j in 1:len_shape) ### 파라미터: shape
114      {
115        color_counter_init = color_counter
116        legend_name = NULL;
117        plot(x, sgamma3(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
118          =2, type = 'n', main="Survival Function")
119        for (k in 1:len_scale) ### 파라미터: scale
120        {
121          lines(x, sgamma3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
122          color_counter = color_counter + 1;
123          legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
124            sep=""))
125        }
126        legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
127      }
128    } else if (func == "hgammma3") # 위험함수
129    {
130      for (j in 1:len_shape) ### 파라미터: shape
131      {
132        color_counter_init = color_counter
133        legend_name = NULL;
134        plot(x, hgammma3(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
135          =2, type = 'n', main="Hazard Function")
136        for (k in 1:len_scale) ### 파라미터: scale
137        {
138          lines(x, hgammma3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
139          color_counter = color_counter + 1;
140          legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
141            sep=""))
142        }
143      }
144    }
145  }
146}

```

```

139         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
140     }
141   }
142 }
143
144
145 par(mfrow = c(9, 8))
146 plot.gamma3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dgamma3")
147
148 par(mfrow = c(9, 8))
149 plot.gamma3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pgamma3")
150
151 par(mfrow = c(9, 8))
152 plot.gamma3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sgamma3")
153
154 par(mfrow = c(9, 8))
155 plot.gamma3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hgamma3")

```

시스템의 수명이 일정기간 동안 보장되는 경우에는, 3모수 감마분포를 적용하는 것이 보다 적절하다.

척도 모수(scale parameter)는 종종 $\lambda = \frac{1}{\theta}$ 로 표시되는 경우도 있다.

위치모수 $\gamma = 0$ 이면 2모수 감마 분포가 된다.

14.14.3 Log-gamma Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{e^{c\frac{t-a}{b}} - e^{\frac{t-a}{b}}}{b\Gamma(c)}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\frac{\gamma \left[c, e^{\frac{t-a}{b}} \right]}{\Gamma(c)}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	IHR
변수		$t \in \mathbf{R}$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 27: Log-gamma 분포함수에 기반한 척도 함수

14.14.4 Generalized Gamma Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{d(t-a)^{cd-1}}{b^{cd}\Gamma(c)}e^{-\left(\frac{t-a}{b}\right)^d}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\frac{\gamma\left[c, \left(\frac{t-a}{b}\right)^d\right]}{\Gamma(c)}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{d(t-a)^{cd-1}e^{-\left(\frac{t-a}{b}\right)^d}}{b^{cd}\gamma\left[c, \left(\frac{t-a}{b}\right)^d\right]}$	
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0, d > 0$	

Table 28: Generalized Gamma 분포함수에 기반한 척도 함수

Stacy(1962)[60]는 파라메터가 3개인 Gamma 분포에서, 파라메터가 4개인 위의 generalized gamma 분포를 유도하였다.

14.15 Generalized Exponential Geometric Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c(1-p)e^{-\frac{t-a}{b}} \left[1-e^{-\frac{t-a}{b}}\right]^{c-1}}{b \left[1-pe^{-\frac{t-a}{b}}\right]^{c-1}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \left[\frac{1-e^{-\frac{t-a}{b}}}{1-pe^{-\frac{t-a}{b}}} \right]^c$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c(1-p)e^{-\frac{t-a}{b}} \left[1-e^{-\frac{t-a}{b}}\right]^{c-1}}{b \left[1-pe^{-\frac{t-a}{b}}\right]^{c+1} - \left[1-pe^{-\frac{t-a}{b}}\right]^c \left[1-e^{-\frac{t-a}{b}}\right]^c}$	IDHR for $p \in \left(\frac{c-1}{c+1}, 1\right)$ and $c > 1$ IHR for $p \in \left(0, \frac{c-1}{c+1}\right)$ and $c > 1$ DHR for otherwise.
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0, p \in (0, 1)$	

Table 29: Generalized Exponential 분포함수에 기반한 척도 함수

14.16 Gompertz Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\alpha e^{\beta t} e^{\frac{\alpha}{\beta}} [1 - e^{\beta t}]$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{\frac{\alpha}{\beta}} [1 - e^{\beta t}]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\alpha e^{\beta t}$	IHR
변수		$t \geq 0$	
파라메터		$\alpha \geq 0, \beta > 0$	

Table 30: Gompertz 분포함수에 기반한 척도 함수

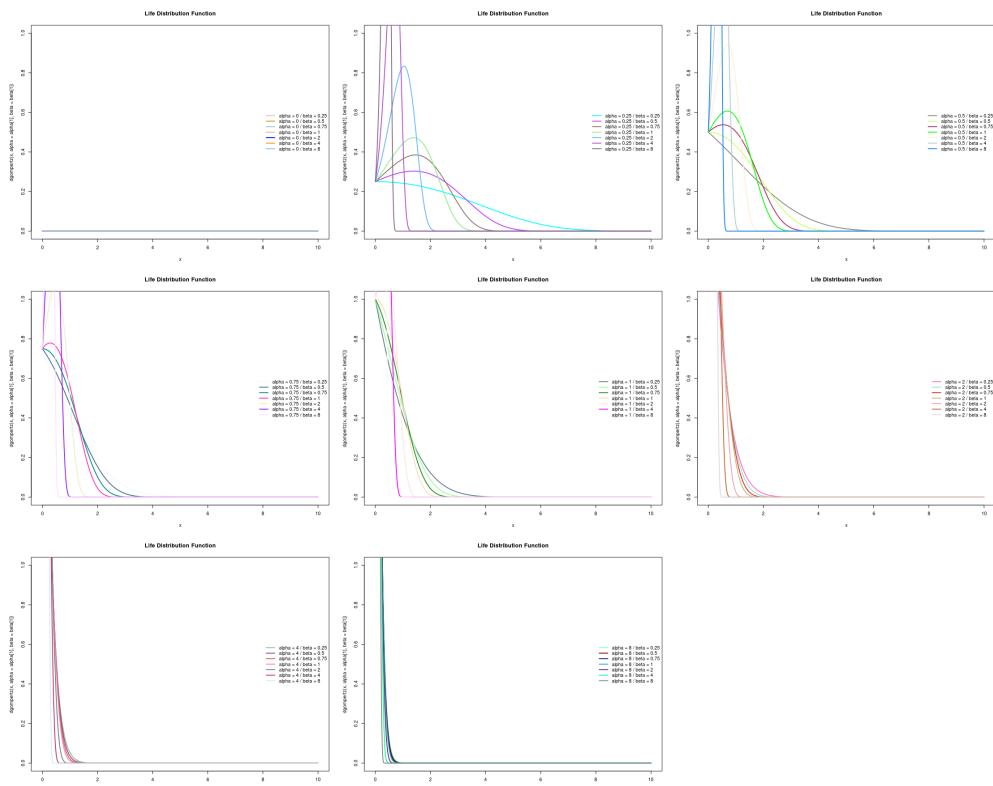


Figure 66: Gompertz Distribution에 기반한 수명분포함수

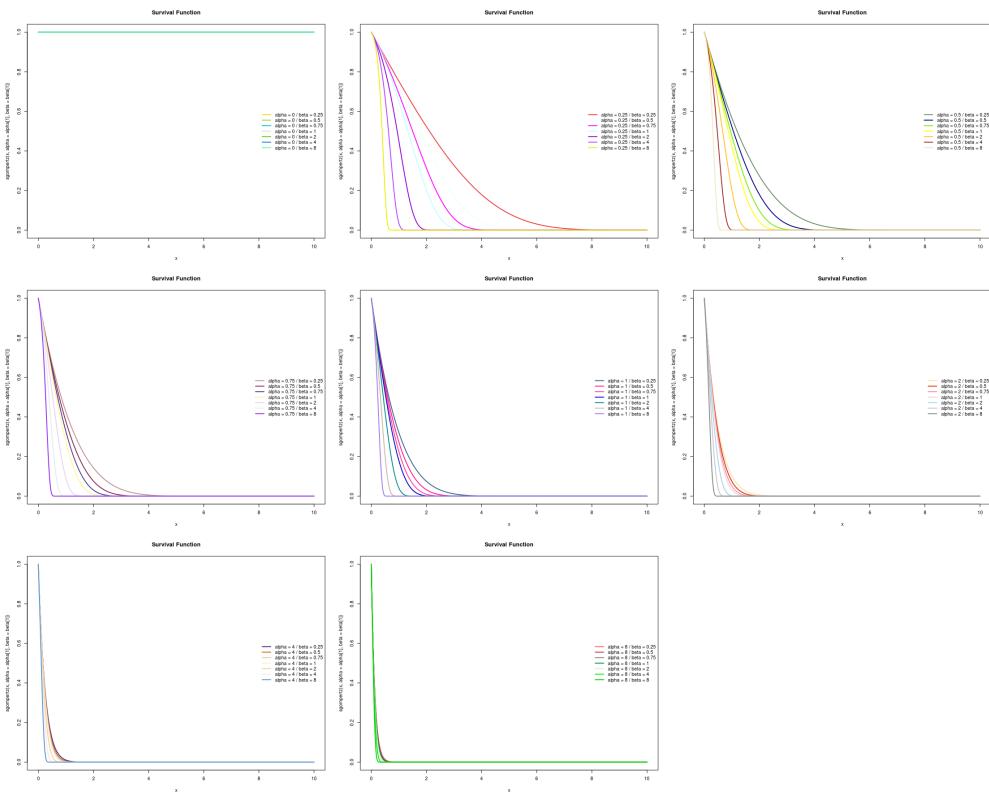


Figure 67: Gompertz Distribution에 기반한 생존함수

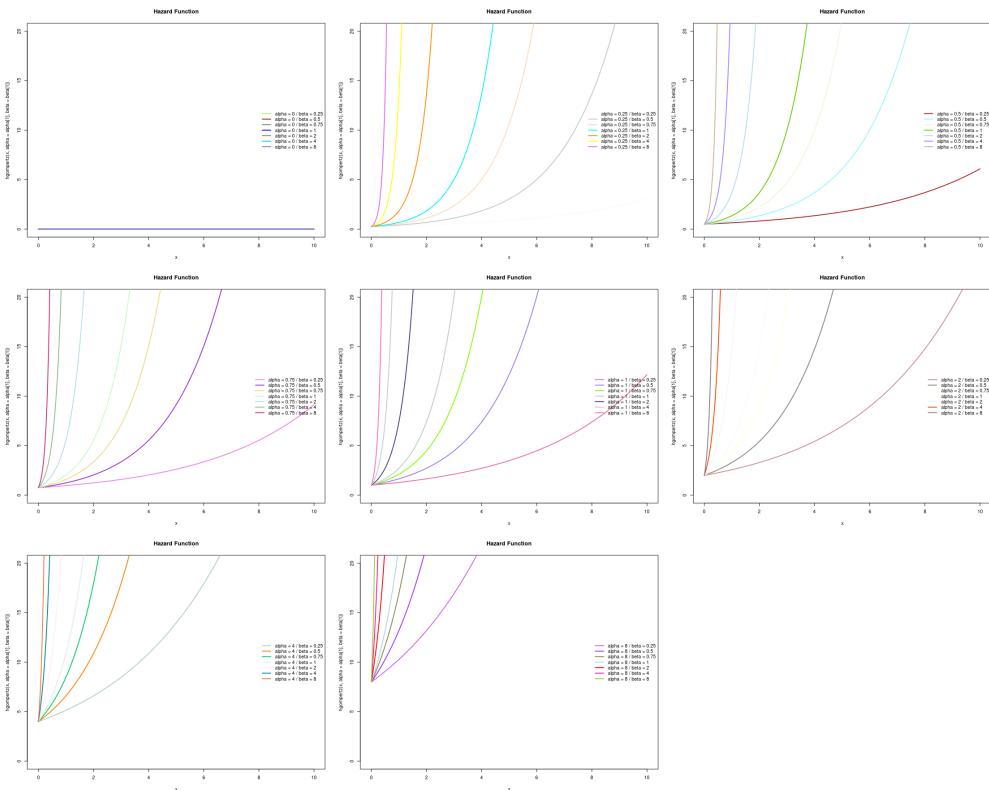


Figure 68: Gompertz Distribution에 기반한 위험함수

Code 25. Gompertz Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Gompertz Distribution
6 ### parameter
7 alpha = c(0, 0.25, 0.5, 0.75, 1, 2, 4, 8)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dgompertz = function (x, alpha = 0, beta = 1)
16 {
17   fx = alpha * exp(beta * x) * exp(alpha/beta * (1 - exp(beta * x)))
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 pgompertz = function (x, alpha = 0, beta = 1)
24 {
25   fx = -(sgompertz(x, alpha, beta) - 1)
26   return(fx)
27 }
28
29
30 ### 생존함수
31 sgompertz = function (x, alpha = 0, beta = 1)
32 {
33   fx = exp(alpha/beta * (1 - exp(beta * x)))
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hgompertz = function (x, alpha = 0, beta = 1)
40 {
41   fx = dgompertz(x, alpha, beta) / sgompertz(x, alpha, beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.gompertz_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dgompertz")
51 {
52   color=colorPalette(300)
53
54   len_alpha = length(alpha) # alpha 파라메터의 길이
55   len_beta = length(beta) # beta 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_alpha) ### 파라메터: alpha
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dgompertz") # 수명분포
64     {
65       plot(x, dgompertz(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
66           Distribution Function")
67       for (j in 1:len_beta) ### 파라메터: beta
68       {
69         lines(x, dgompertz(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
72       }
73     }
74     else if (func == "pgompertz") # 누적분포함수
75     {
76       plot(x, pgompertz(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
77           Cumulative Distribution Function")
78       for (j in 1:len_beta) ### 파라메터: beta
79       {
80         lines(x, pgompertz(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
83     }
84     else if (func == "sgompertz") # 생존함수
85     {
86       plot(x, sgompertz(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
87           Survival Function")
88     }
89   }
90 }
```

```

86     for (j in 1:len_beta) ### 파라미터: beta
87     {
88       lines(x, sgompertz(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
91     }
92   }
93 else if (func == "hgompertz") # 위험함수
94 {
95   plot(x, hgompertz(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
96   Function")
97   for (j in 1:len_beta) ### 파라미터: beta
98   {
99     lines(x, hgompertz(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102  }
103  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.gompertz_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="dgompertz")
109
110 par(mfrow = c(3, 3))
111 plot.gompertz_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pgompertz")
112
113 par(mfrow = c(3, 3))
114 plot.gompertz_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sgompertz")
115
116 par(mfrow = c(3, 3))
117 plot.gompertz_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 20), func="hgompertz")

```

14.17 Gompertz-Makeham Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\zeta \left[\frac{\alpha}{\beta} (1 - e^{\beta t}) - \zeta t \right] + \alpha e^{\beta t} e^{\frac{\alpha}{\beta} (1 - e^{\beta t}) - \zeta t}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{\frac{\alpha}{\beta} (1 - e^{\beta t}) - \zeta t}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\zeta + \alpha e^{\beta t}$	IHR
변수		$t \geq 0$	
파라메터		$\alpha > 0, \beta > 0, \zeta > 0$	

Table 31: Gompertz-Makeham 분포함수에 기반한 척도 함수



Figure 69: Gompertz-Makeham Distribution에 기반한 수명분포함수



Figure 70: Gompertz-Makeham Distribution에 기반한 생존함수



Figure 71: Gompertz-Makeham Distribution에 기반한 위험함수

Code 26. Gompertz-Makeham Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Gompertz-Makeham Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0, 0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0, 0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 1, length.out = 1000)
13
14
15 ### 수명 분포
16 dgompertzmakeham = function(x, alpha = 1, beta = 1, gamma = 1)
17 {
18   fx = (gamma * ((alpha / beta) * (1 - exp(beta * x)) - gamma * x)) + (alpha * exp(beta * x) * exp((alpha/beta) * (1-exp(beta * x))) - gamma * x))
19   return(fx)
20 }
21
22
23 ### 누적분포함수
24 pgompertzmakeham = function(x, alpha = 1, beta = 1, gamma = 1)
25 {
26   fx = -(sgompertzmakeham(x, alpha, beta, gamma) - 1)
27   return(fx)
28 }
29
30
31 ### 생존함수
32 sgompertzmakeham = function (x, alpha = 1, beta = 1, gamma = 1)
33 {
34   fx = exp((alpha/beta) * (1-exp(beta * x))) - gamma * x
35   return(fx)
36 }
37
38
39 ### 위험함수
40 hgompertzmakeham = function (x, alpha = 1, beta = 1, gamma = 1)
41 {
42   fx = dgompertzmakeham(x, alpha, beta, gamma) / sgompertzmakeham(x, alpha, beta, gamma)
43   return(fx)
44 }
45
46
47
48
49
50 ##### Plot
51 plot.gompertzmakeham_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dgompertzmakeham")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라미터의 길이
56   len_beta = length(beta) # beta 파라미터의 길이
57   len_gamma = length(gamma) # gamma 파라미터의 길이
58
59   color_counter = 1
60   for (i in 1:len_alpha) ### 파라미터: alpha
61   {
62     if (func=="dgompertzmakeham") # 수명분포
63     {
64       for (j in 1:len_beta) ### 파라미터: beta
65       {
66         color_counter_init = color_counter
67         legend_name = NULL;
68         plot(x, dgompertzmakeham(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
69         for (k in 1:len_gamma) ### 파라미터: gamma
70         {
71           lines(x, dgompertzmakeham(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
72           color_counter = color_counter + 1;
73           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
74         }
75         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
76       }
77     }
78   else if (func == "pgompertzmakeham") # 누적분포함수
79   {
80     for (j in 1:len_beta) ### 파라미터: beta
81     {
82       color_counter_init = color_counter
83       legend_name = NULL;

```

```

84   plot(x, pgompertzmakeham(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
85     =2, type = 'n', main="Cumulative Distribution Function")
86   for (k in 1:len_gamma) ### 파라미터: gamma
87   {
88     lines(x, pgompertzmakeham(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
89     color_counter = color_counter + 1;
90     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
91   }
92   legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
93 }
94 else if (func == "sgompertzmakeham") # 생존함수
95 {
96   for (j in 1:len_beta) ### 파라미터: beta
97   {
98     color_counter_init = color_counter
99     legend_name = NULL;
100    plot(x, sgompertzmakeham(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
101      =2, type = 'n', main="Survival Function")
102    for (k in 1:len_gamma) ### 파라미터: gamma
103    {
104      lines(x, sgompertzmakeham(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
105      color_counter = color_counter + 1;
106      legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
107    }
108    legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
109  }
110 else if (func == "hgompertzmakeham") # 위험함수
111 {
112   for (j in 1:len_beta) ### 파라미터: beta
113   {
114     color_counter_init = color_counter
115     legend_name = NULL;
116     plot(x, hgompertzmakeham(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
117       =2, type = 'n', main="Hazard Function")
118     for (k in 1:len_gamma) ### 파라미터: gamma
119     {
120       lines(x, hgompertzmakeham(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
121       color_counter = color_counter + 1;
122       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
123     }
124     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
125   }
126 }
127 }
128 par(mfrow = c(9, 8))
129 plot.gompertzmakeham_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dgompertzmakeham")
130
131 par(mfrow = c(9, 8))
132 plot.gompertzmakeham_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pgompertzmakeham")
133
134 par(mfrow = c(9, 8))
135 plot.gompertzmakeham_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sgompertzmakeham")
136
137 par(mfrow = c(9, 8))
138 plot.gompertzmakeham_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hgompertzmakeham")
139

```

14.18 Hjorth Distribution(호스 분포)

Table 32: 호스 분포함수에 기반한 척도 함수

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{(1+\beta t)\delta t^2 + \theta}{(1+\beta t)^\beta} e^{-\frac{1}{2}\delta t^2}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\frac{e^{-\frac{1}{2}\delta t^2}}{(1+\beta t)^\beta}$	
위험률 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\delta t + \frac{\theta}{1+\beta t}$	
변수		$t \geq 0$	
파라미터		$\delta \geq 0, \beta \geq 0, \theta \geq 0$	



Figure 72: Hjorth Distribution에 기반한 수명분포함수



Figure 73: Hjorth Distribution에 기반한 생존함수

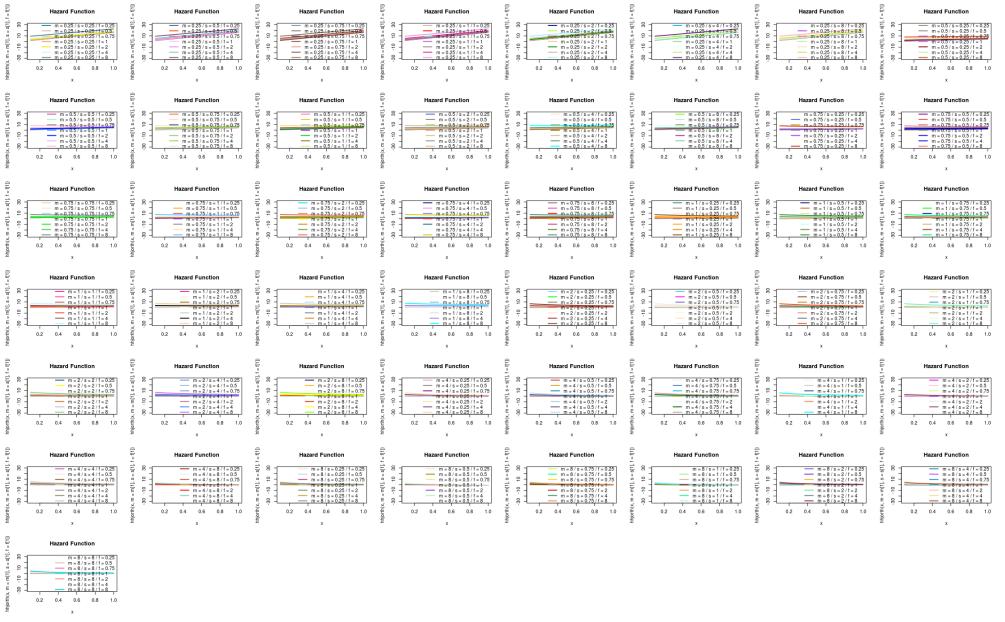


Figure 74: Hjorth Distribution에 기반한 위험함수

Code 27. Hjorth Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(rmutil)
4
5 ###### Hjorth Distribution
6 #### parameter
7 delta = c(0.25, 0.5, 0.75, 1, 2, 4, 8) # m (delta)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8) # s (beta)
9 theta = c(0.25, 0.5, 0.75, 1, 2, 4, 8) # f (theta)
10
11 #### input varialbe
12 x = seq(0.1, 1, length.out = 1000)
13
14
15 #### 수명 분포
16 dhjorth(x, m = 1, s = 1, f = 1)
17
18
19 #### 분위수 함수
20 qhjorth(x, m = 1, s = 1, f = 1)
21
22
23 #### 난수 함수
24 rhjorth(x, m = 1, s = 1, f = 1)
25
26
27 #### 누적분포함수
28 phjorth(x, m = 1, s = 1, f = 1)
29
30
31 #### 생존함수
32 shjorth = function (x, m = 1, s = 1, f=1)
33 {
34   fx = 1 - phjorth(x, m = m, s = m, f = f)
35   return(fx)
36 }
37
38
39 #### 위험함수
40 hhjorth = function (x, m = 1, s = 1, f=1)
41 {
42   fx = dhjorth(x, m = m, s = m, f = f) / shjorth(x, m = m, s = m, f = f)
43   return(fx)
44 }
45
46
47
48
49
50 ###### Plot
51 plot.hjorth_seq = function(x, m = 1, s = 1, f = 1, xlim=c(0, 10), ylim=c(0, 5), func="dhjorth")
52 {
53   color=colorPalette(300)
54
55   len_m = length(m) # m 파라메터의 길이
56   len_s = length(s) # s 파라메터의 길이
57   len_f = length(f) # f 파라메터의 길이
58
59   color_counter = 1
60   for (i in 1:len_m) ### 파라메터: m
61   {
62     if (func=="dhjorth") # 수명분포
63     {
64       for (j in 1:len_s) ### 파라메터: s
65       {
66         color_counter_init = color_counter
67         legend_name = NULL;
68         plot(x, dhjorth(x, m=m[i], s=s[j], f=f[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
69           Distribution Function")
70         for (k in 1:len_f) ### 파라메터: f
71         {
72           lines(x, dhjorth(x, m=m[i], s=s[j], f=f[k]), col=color[color_counter], lwd=2);
73           color_counter = color_counter + 1;
74           legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], " / f = ", f[k], sep=""))
75         }
76         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
77       }
78     }
79   else if (func == "phjorth") # 누적분포함수
80   {
81     for (j in 1:len_s) ### 파라메터: s
82     {
83       color_counter_init = color_counter
84       legend_name = NULL;
85       plot(x, phjorth(x, m=m[i], s=s[j], f=f[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
86           Distribution Function")
87       for (k in 1:len_f) ### 파라메터: f

```

```

87      }
88      lines(x, phjorth(x, m=m[i], s=s[j], f=f[k]), col=color[color_counter], lwd=2);
89      color_counter = color_counter + 1;
90      legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], " / f = ", f[k], sep=""))
91    }
92    legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
93  }
94
95  else if (func == "shjorth") # 생존함수
96  {
97    for (j in 1:len_s) ### 파라메터: s
98    {
99      color_counter_init = color_counter
100     legend_name = NULL;
101     plot(x, shjorth(x, m=m[1], s=s[1], f=f[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival
102       Function")
103     for (k in 1:len_f) ### 파라메터: f
104     {
105       lines(x, shjorth(x, m=m[i], s=s[j], f=f[k]), col=color[color_counter], lwd=2);
106       color_counter = color_counter + 1;
107       legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], " / f = ", f[k], sep=""))
108     }
109     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
110   }
111
112  else if (func == "hhjorth") # 위험함수
113  {
114    for (j in 1:len_s) ### 파라메터: s
115    {
116      color_counter_init = color_counter
117      legend_name = NULL;
118      plot(x, hhjorth(x, m=m[1], s=s[1], f=f[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
119       Function")
120      for (k in 1:len_f) ### 파라메터: f
121      {
122        lines(x, hhjorth(x, m=m[i], s=s[j], f=f[k]), col=color[color_counter], lwd=2);
123        color_counter = color_counter + 1;
124        legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], " / f = ", f[k], sep=""))
125      }
126    }
127  }
128}
129
130 par(mfrow = c(9, 8))
131 plot.hjorth_seq(x, delta, beta, theta, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dhjorth")
132
133 par(mfrow = c(9, 8))
134 plot.hjorth_seq(x, delta, beta, theta, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="phjorth")
135
136 par(mfrow = c(9, 8))
137 plot.hjorth_seq(x, delta, beta, theta, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="shjorth")
138
139 par(mfrow = c(9, 8))
140 plot.hjorth_seq(x, delta, beta, theta, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hhjorth")

```

호스 분포는 3개의 모수를 가진 분포로, 모수들의 값에 따라 위험률 함수(고장률 함수)가 증가, 감소, 육조 모양을 가지는 매우 유동적인 수명 분포이다.

- $\theta = 0$: 레일리 분포
- $\theta = \beta = 0$: 지수 분포
- $\delta = 0$: 위험률(고장률)이 감소하는 분포(DFR)
- $\delta \geq \theta\beta$: 위험률(고장률)이 증가하는 분포(IFR)
- $0 < \delta < \theta\beta$: 위험률 함수(고장률 함수)가 육조 모양인 분포

14.19 Hyperbolic Secant Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b\pi} \operatorname{sech}\left(\frac{t-a}{b}\right)$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \frac{2}{\pi} \arctan\left(e^{\frac{t-a}{b}}\right)$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{\operatorname{sech}\left(\frac{t-a}{b}\right)}{b \left[\pi - 2 \arctan\left(e^{\frac{t-a}{b}}\right) \right]}$	IHR with $\lim_{t \rightarrow \infty} h(t) = \frac{1}{b}$
변수		$t \in \mathbf{R}$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 33: Hyperbolic Secant 분포함수에 기반한 척도 함수

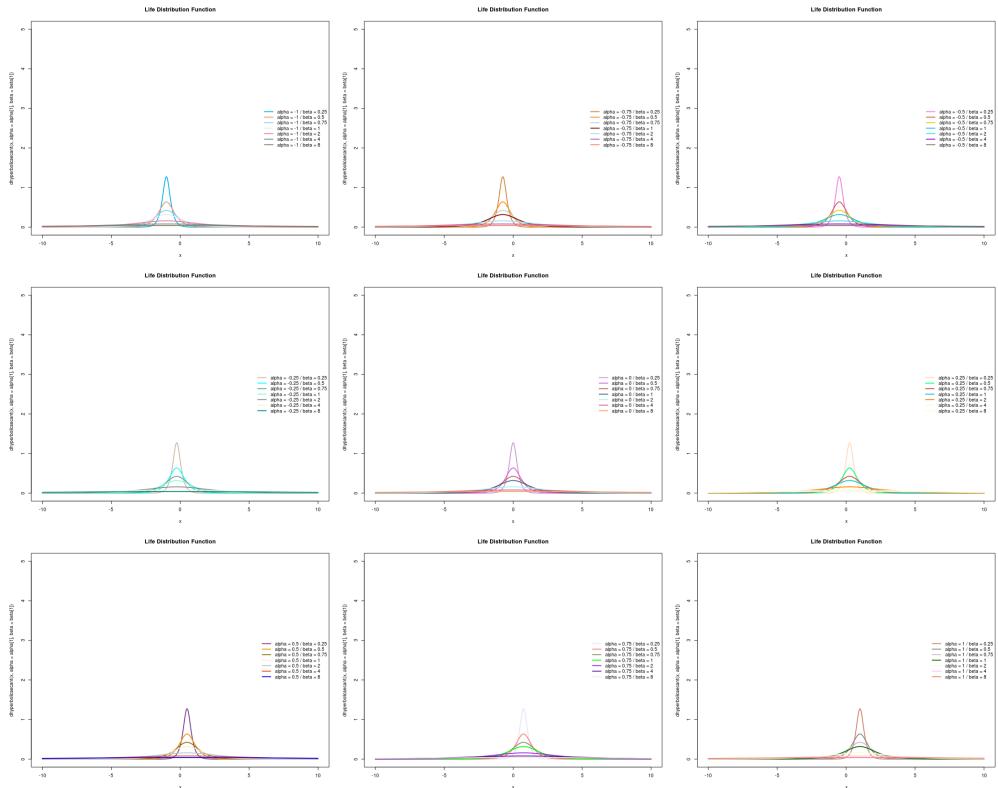


Figure 75: Hyperbolic Secant Distribution에 기반한 수명분포함수

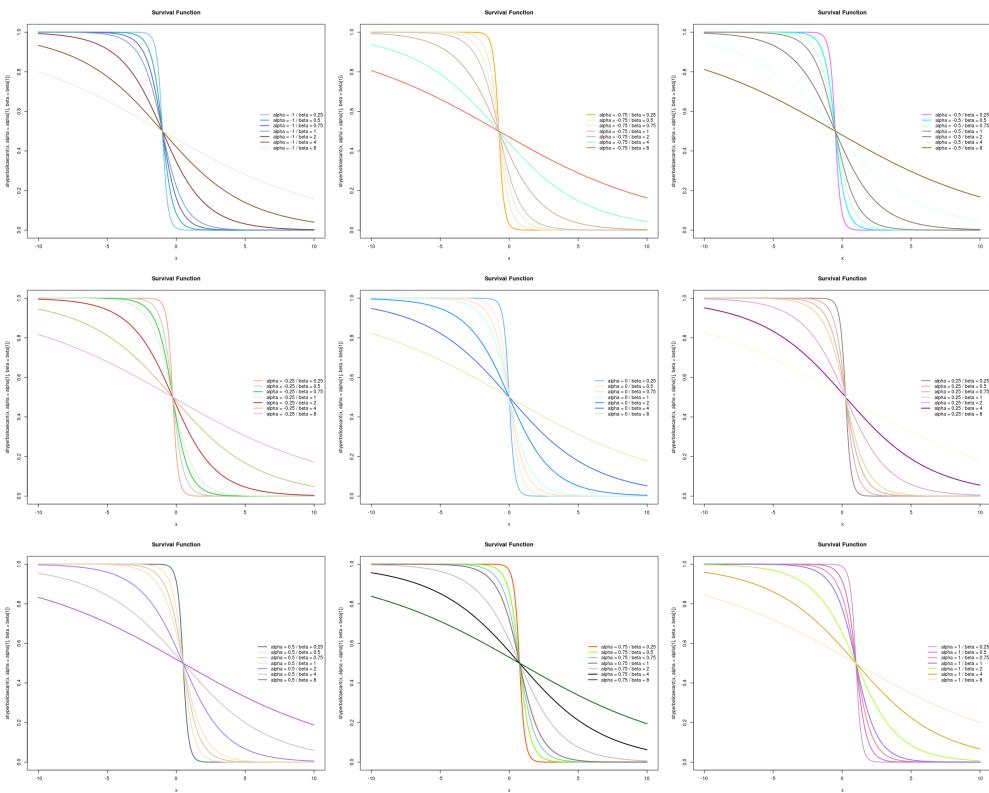


Figure 76: Hyperbolic Secant Distribution에 기반한 생존함수

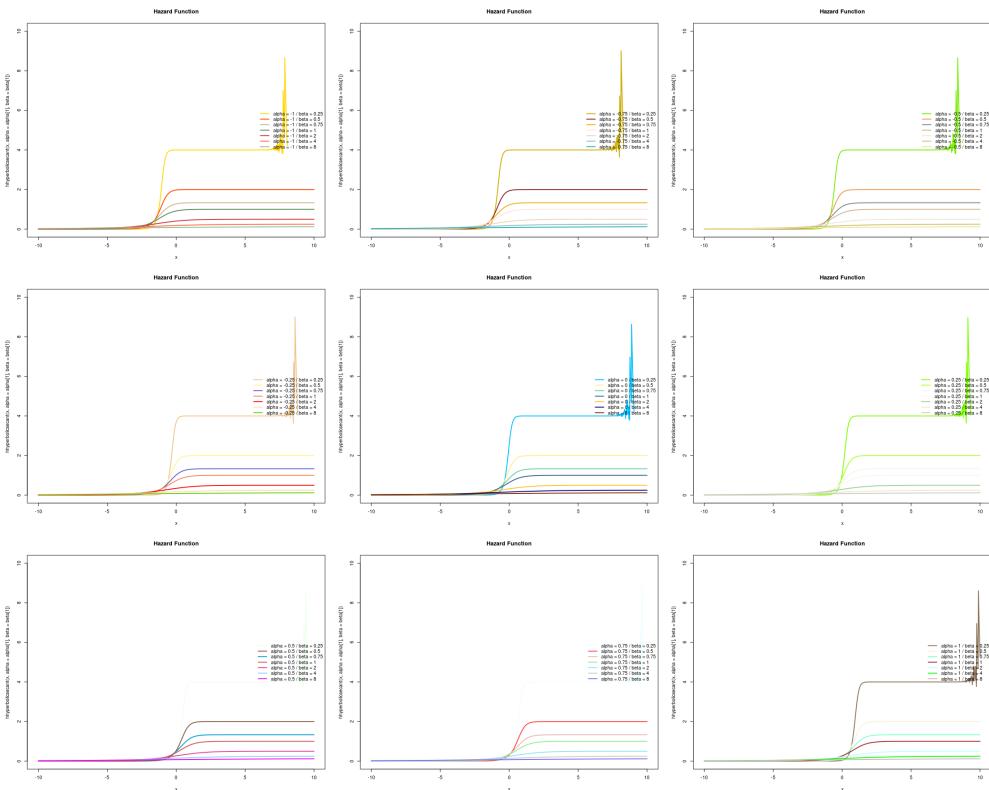


Figure 77: Hyperbolic Secant Distribution에 기반한 위험함수

Code 28. Hyperbolic Secant Distribution Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 # https://www.rdocumentation.org/packages/VaRES/versions/1.0/topics/secant
2
3 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
4 source("colorPalette.R")
5 require(pracma)
6
7
8 ##### hyperbolicsecant Distribution
9 ### parameter
10 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
11 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
12
13 ### input varialbe
14 x = seq(-10, 10, length.out = 1000)
15
16
17 ### 수명 분포
18 dhyperbolicsecant = function(x, alpha = 1, beta = 1)
19 {
20   fx = (1 / (beta * pi)) * sech((x - alpha)/beta)
21   return(fx)
22 }
23
24
25 ### 누적분포함수
26 phyperbolicsecant = function(x, alpha = 1, beta = 1)
27 {
28   fx = -(shyperbolicsecant(x, alpha, beta) - 1)
29   return(fx)
30 }
31
32
33 ### 생존함수
34 shyperbolicsecant = function (x, alpha = 1, beta = 0)
35 {
36   fx = 1 - (2/pi) * atan(exp((x - alpha)/beta))
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hhyperbolicsecant = function (x, alpha = 1, beta = 0)
43 {
44   fx = dhyperbolicsecant(x, alpha, beta) / shyperbolicsecant(x, alpha, beta)
45   return(fx)
46 }
47
48
49
50
51
52 ##### Plot
53 plot.hyperbolicsecant_seq = function(x, alpha = 1, beta = 0, xlim=c(0, 10), ylim=c(0, 5), func="dhyperbolicsecant")
54 {
55   color=colorPalette(300)
56
57   len_alpha = length(alpha) # alpha 파라메터의 길이
58   len_beta = length(beta) # beta 파라메터의 길이
59
60   color_counter = 1
61   for (i in 1:len_alpha) ### 파라메터: alpha
62   {
63     color_counter_init = color_counter
64     legend_name = NULL;
65
66     if (func=="dhyperbolicsecant") # 수명분포
67     {
68       plot(x, dhyperbolicsecant(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
69         ="Life Distribution Function")
70       for (j in 1:len_beta) ### 파라메터: beta
71       {
72         lines(x, dhyperbolicsecant(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
73         color_counter = color_counter + 1;
74         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
75       }
76     }
77     else if (func == "phyperbolicsecant") # 누적분포함수
78     {
79       plot(x, phyperbolicsecant(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
80         ="Cumulative Distribution Function")
81       for (j in 1:len_beta) ### 파라메터: beta
82       {
83         lines(x, phyperbolicsecant(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
84         color_counter = color_counter + 1;
85         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
86     }
87   }
88 }
```

```

85 }
86 else if (func == "shyperbolicsecant") # 생존함수
87 {
88   plot(x, shyperbolicsecant(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
89     ="Survival Function")
90   for (j in 1:len_beta) ### 파라미터: beta
91   {
92     lines(x, shyperbolicsecant(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
93     color_counter = color_counter + 1;
94     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
95   }
96 else if (func == "hhyperbolicsecant") # 위험함수
97 {
98   plot(x, hhyperbolicsecant(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
99     ="Hazard Function")
100  for (j in 1:len_beta) ### 파라미터: beta
101  {
102    lines(x, hhyperbolicsecant(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
103    color_counter = color_counter + 1;
104    legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
105  }
106  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
107 }
108 }
109 par(mfrow = c(3, 3))
110 plot.hyperbolicsecant_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dhyperbolicsecant")
111 par(mfrow = c(3, 3))
112 plot.hyperbolicsecant_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="phyperbolicsecant")
113 par(mfrow = c(3, 3))
114 plot.hyperbolicsecant_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="shyperbolicsecant")
115 par(mfrow = c(3, 3))
116 plot.hyperbolicsecant_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hhyperbolicsecant")
117 par(mfrow = c(3, 3))
118 plot.hyperbolicsecant_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hhyperbolicsecant")
119 par(mfrow = c(3, 3))
120 plot.hyperbolicsecant_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hhyperbolicsecant")

```

14.20 Laplace Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{2b} e^{-\frac{ t-a }{b}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \frac{1}{2} \left[1 + \text{sign}(t-a) \left(1 - e^{-\frac{ t-a }{b}} \right) \right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{e^{-\frac{ t-a }{b}}}{b \left[2 - \left(1 + \text{sign}(t-a) \left(1 - e^{-\frac{ t-a }{b}} \right) \right) \right]}$	
변수		$t \in \mathbf{R}$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 34: Laplace 분포함수에 기반한 척도 함수

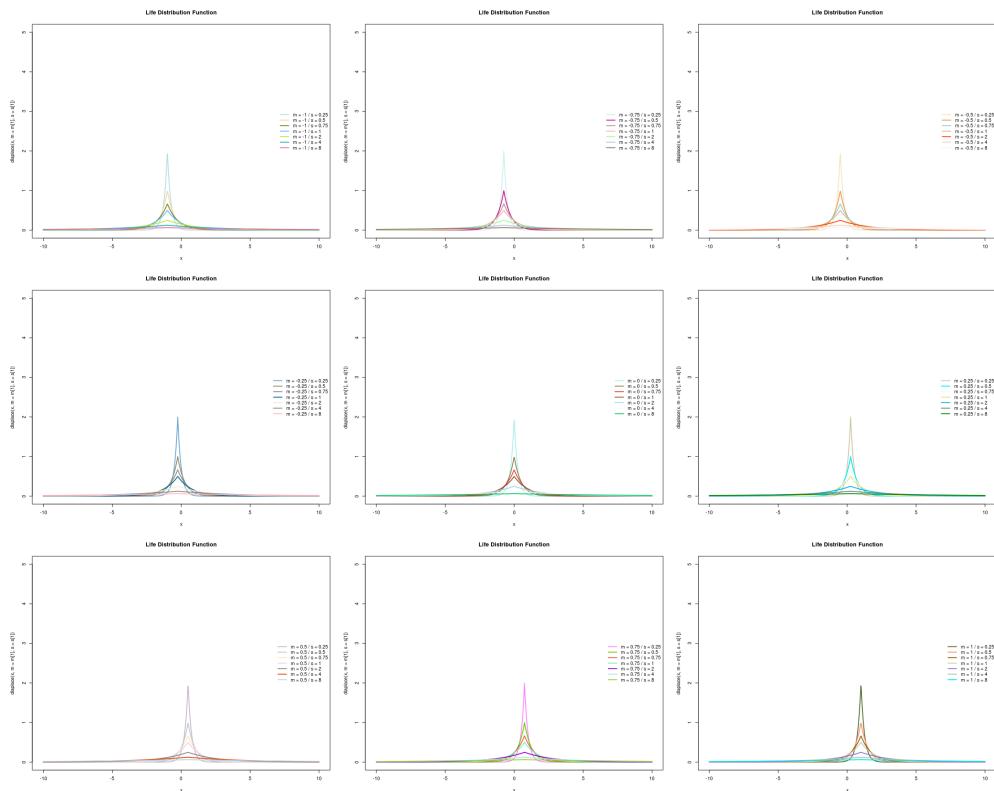


Figure 78: Laplace Distribution에 기반한 수명분포함수

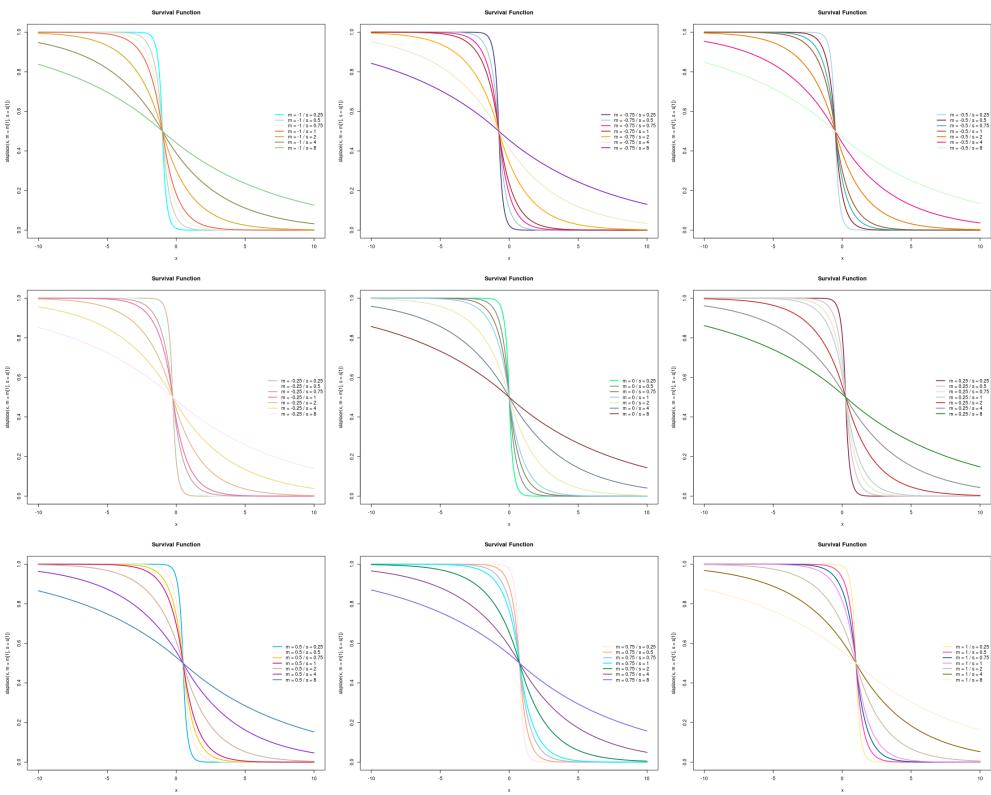


Figure 79: Laplace Distribution에 기반한 생존함수

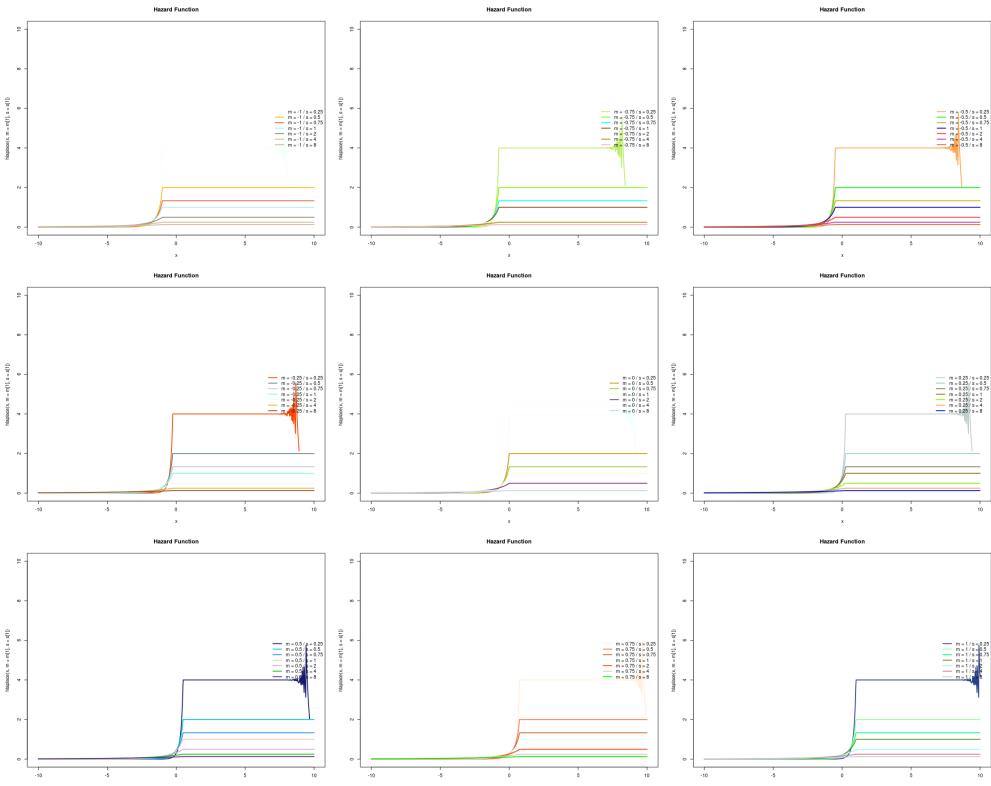


Figure 80: Laplace Distribution에 기반한 위험함수

Code 29. Laplace Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(rmutil)
4
5 ###### laplace Distribution
6 #### parameter
7 m = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 s = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 #### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 #### 수명 분포
15 dlaplace(x, m = 0, s = 1)
16
17
18 #### 분위수 함수
19 qlaplace(x, m = 0, s = 1)
20
21
22 #### 난수 함수
23 rlaplace(x, m = 0, s = 1)
24
25
26 #### 누적분포함수
27 plaplace(x, m = 0, s = 1)
28
29
30 #### 생존함수
31 slaplace = function(x, m = 0, s = 1)
32 {
33   fx = 1 - plaplace(x, m = m, s = s)
34   return(fx)
35 }
36
37
38 #### 위험함수
39 hlaplace = function(x, m = 0, s = 1)
40 {
41   fx = dlaplace(x, m = m, s = s) / slaplace(x, m = m, s = s)
42   return(fx)
43 }
44
45
46
47
48
49 ###### Plot
50 plot.laplace_seq = function(x, m = 0, s = 1, xlim=c(0, 10), ylim=c(0, 5), func="dlaplace")
51 {
52   color=colorPalette(300)
53
54   len_m = length(m) # m 파라메터의 길이
55   len_s = length(s) # s 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_m) ### 파라메터: m
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dlaplace") # 수명분포
64     {
65       plot(x, dlaplace(x, m=m[i], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution
66           Function")
67       for (j in 1:len_s) ### 파라메터: s
68       {
69         lines(x, dlaplace(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
72       }
73     }
74     else if (func == "plaplace") # 누적분포함수
75     {
76       plot(x, plaplace(x, m=m[1], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
77           Distribution Function")
78       for (j in 1:len_s) ### 파라메터: s
79       {
80         lines(x, plaplace(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
83       }
84     }
85     else if (func == "slaplace") # 생존함수
86     {
87       plot(x, slaplace(x, m=m[1], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
88     }
89   }
90 }
```

```

87     for (j in 1:len_s) ### 파라미터: s
88     {
89       lines(x, slaplace(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
90       color_counter = color_counter + 1;
91       legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
92     }
93   }
94 else if (func == "hlaplace") # 위험함수
95 {
96   plot(x, hlaplace(x, m=m[1], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
97   for (j in 1:len_s) ### 파라미터: s
98   {
99     lines(x, hlaplace(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
102  }
103 }
104 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
105 }
106 }
107
108 par(mfrow = c(3, 3))
109 plot.laplace_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dlaplace")
110
111 par(mfrow = c(3, 3))
112 plot.laplace_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 1), func="plaplace")
113
114 par(mfrow = c(3, 3))
115 plot.laplace_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 1), func="slaplace")
116
117 par(mfrow = c(3, 3))
118 plot.laplace_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hlaplace")

```

14.20.1 Log-Laplace Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\begin{cases} \frac{1}{b} \frac{cd}{c+d} \left(\frac{t}{b}\right)^{c-1} & \text{for } 0 \leq t \leq b \\ \frac{1}{b} \frac{cd}{c+d} \left(\frac{b}{t}\right)^{d+1} & \text{for } t \geq b \end{cases}$	
생존함수 (신뢰도함수)	$\begin{aligned} S(t) &= P(T > t) \\ &= 1 - P(T \leq t) \\ &= 1 - F(t) \\ &= e^{-H(t)} \end{aligned}$	$\begin{cases} 1 - \frac{d}{c+d} \left(\frac{t}{b}\right)^c & \text{for } 0 \leq t \leq b \\ \frac{c}{c+d} \left(\frac{b}{t}\right)^d & \text{for } t \geq b \end{cases}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\begin{cases} \frac{cd}{b} \left(\frac{t}{b}\right)^{c-1} [c + d - d \left(\frac{t}{b}\right)^c]^{-1} & \text{for } 0 \leq t \leq b \\ \frac{cd}{b} \left(\frac{b}{t}\right)^{d+1} \left[c \left(\frac{b}{t}\right)^d\right]^{-1} & \text{for } t \geq b \end{cases}$	
변수 파라메터		$\begin{aligned} t &\geq 0 \\ b &> 0, c > 0, d > 0 \end{aligned}$	

Table 35: Log-Laplace 분포함수에 기반한 척도 함수

14.21 Linear Hazard Rate Distribution(선형 증가 분포)

Table 36: 선형증가 분포함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$(\alpha t + \beta)e^{-(\frac{1}{2}\alpha t^2 + \beta t)}$
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$e^{-\int_0^t (\alpha u + \beta) du} = e^{-(\frac{1}{2}\alpha t^2 + \beta t)}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\alpha t + \beta$
변수		$t \geq 0$
파라메터		$\alpha \geq 0, \beta > 0$

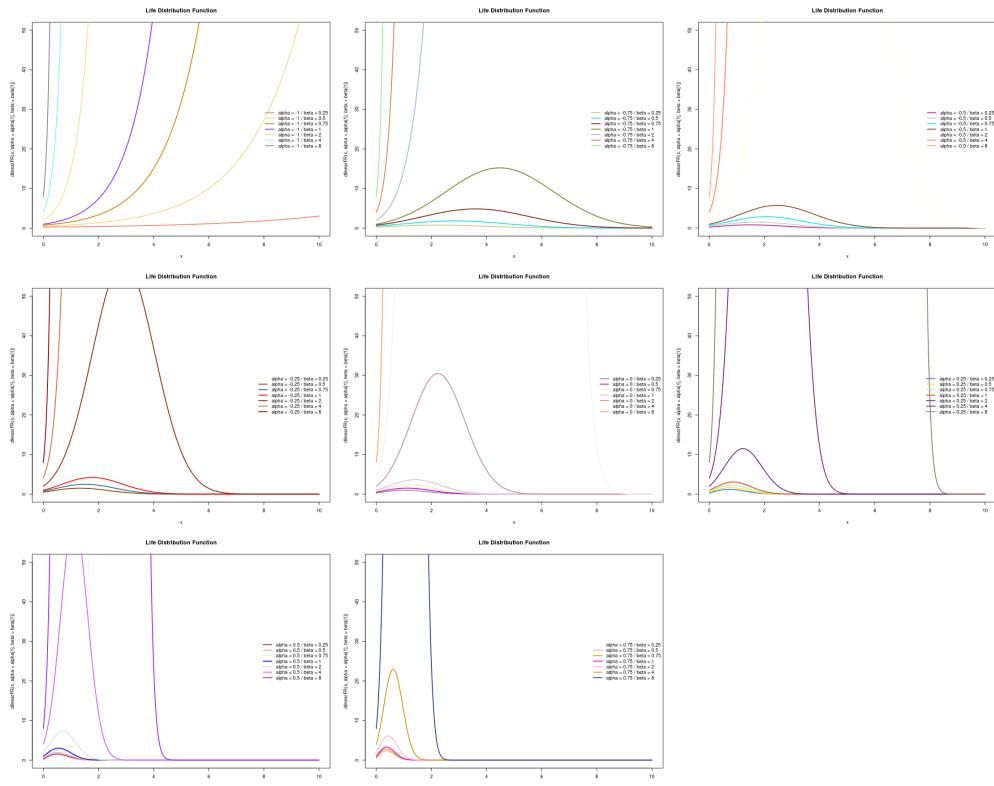


Figure 81: 선형증가 Distribution에 기반한 수명분포함수

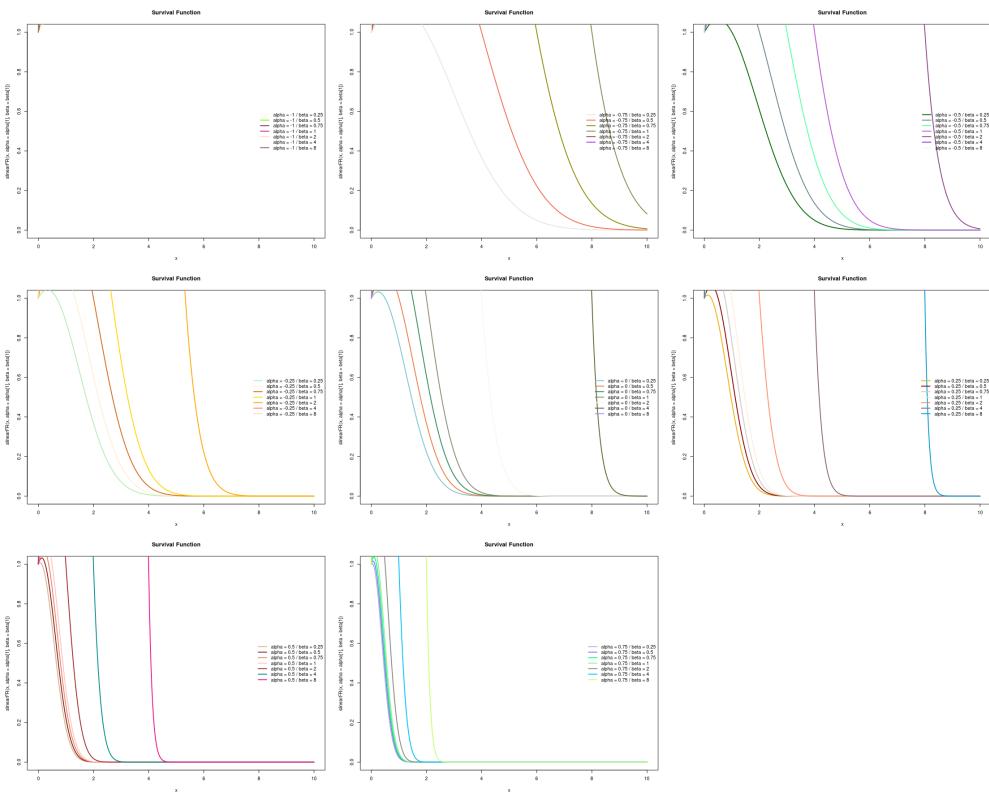


Figure 82: 선형증가 Distribution에 기반한 생존함수

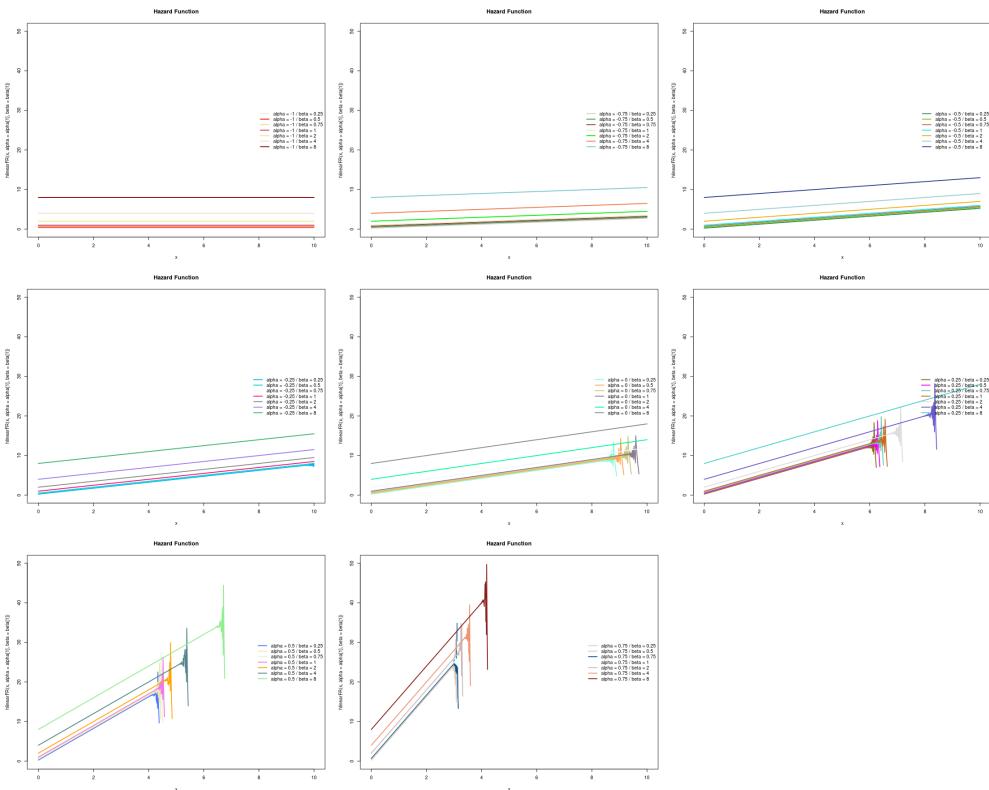


Figure 83: 선형증가 Distribution에 기반한 위험함수

Code 30. 선형증가 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### 선형 증가 분포
6 ### parameter
7 alpha = c(0, 0.25, 0.5, 0.75, 1, 2, 4, 8) # m (delta)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8) # s (beta)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dlinearFR = function(x, alpha = 0, beta = 1)
16 {
17   fx = (alpha * x + beta) * exp(-(1/2) * alpha * x^2 + beta * x)
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 plinearFR = function(x, alpha = 0, beta = 1)
24 {
25   Fx = 1 - exp(-(1/2) * alpha * x^2 + beta * x)
26   return(Fx)
27 }
28
29
30 ### 생존함수
31 slinearFR = function(x, alpha = 0, beta = 1)
32 {
33   fx = 1 - plinearFR(x, alpha = alpha, beta = beta)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hlinearFR = function(x, alpha = 0, beta = 1)
40 {
41   fx = dlinearFR(x, alpha = alpha, beta = beta) / slinearFR(x, alpha = alpha, beta = beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.linearFR_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dlinearFR")
51 {
52   color=colorPalette(300)
53
54   len_alpha = length(alpha) # m 파라미터의 길이
55   len_beta = length(beta) # s 파라미터의 길이
56
57   color_counter = 1
58   for (i in 1:len_alpha) ### 파라미터: m
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dlinearFR") # 수명분포
64     {
65       plot(x, dlinearFR(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
66           Distribution Function")
67       for (j in 1:len_beta) ### 파라미터: s
68       {
69         lines(x, dlinearFR(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
72       }
73     }
74     else if (func == "plinearFR") # 누적분포함수
75     {
76       plot(x, plinearFR(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
77           Cumulative Distribution Function")
78       for (j in 1:len_beta) ### 파라미터: s
79       {
80         lines(x, plinearFR(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
83     }
84     else if (func == "slinearFR") # 생존함수
85     {
86       plot(x, slinearFR(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
87           Survival Function")

```

```

86     for (j in 1:len_beta) ### 파라메터: s
87     {
88       lines(x, slinearFR(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
91     }
92   }
93 else if (func == "hlinearFR") # 위험함수
94 {
95   plot(x, hlinearFR(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
96   Function")
97   for (j in 1:len_beta) ### 파라메터: s
98   {
99     lines(x, hlinearFR(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
102  }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.linearFR_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 50), func="dlinearFR")
109
110 par(mfrow = c(3, 3))
111 plot.linearFR_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="plinearFR")
112
113 par(mfrow = c(3, 3))
114 plot.linearFR_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="slinearFR")
115
116 par(mfrow = c(3, 3))
117 plot.linearFR_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 50), func="hlinearFR")

```

수명을 대표하는 연속확률변수 T 의 위험률 함수(고장률 함수)가 선형증가함수라고 가정한 경우이다.
만일 $\alpha = 0$ 이면, 이 분포는 지수분포가 된다.

만일 $\beta = 0$ 인 경우,

$$\begin{aligned}
h(t) &= \alpha t \\
S(t) &= e^{-\frac{1}{2}\alpha t^2} \\
f(t) &= \alpha t e^{-\frac{1}{2}\alpha t^2} \\
E(t) &= \frac{\sqrt{\pi}}{\sqrt{2\alpha}} \\
Var(t) &= \frac{2}{\alpha} \left(1 - \frac{\pi}{4}\right)
\end{aligned}$$

14.21.1 Generalized Linear Hazard Rate Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$c(\alpha + \beta t) \left[1 - e^{-(\alpha t + \frac{\beta}{2} t^2)}\right]^{c-1} e^{-(\alpha t + \frac{\beta}{2} t^2)}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \left[1 - e^{-(\alpha t + \frac{\beta}{2} t^2)}\right]^c$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c(\alpha + \beta t) e^{-(\alpha t + \frac{\beta}{2} t^2)}}{\left[1 - e^{-(\alpha t + \frac{\beta}{2} t^2)}\right] \left[\left(1 - e^{-(\alpha t + \frac{\beta}{2} t^2)}\right)^{-c} - 1\right]}$	
변수		$t \geq 0$	
파라메터		$\alpha \geq 0, \beta \geq 0$, but not $\alpha = \beta = 0$, $c > 0$	

Table 37: Generalized linear hazard rate 분포함수에 기반한 척도 함수

14.22 Logistic Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} \frac{e^{\frac{t-a}{b}}}{\left[1+e^{\frac{t-a}{b}}\right]^2}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\left[1 + e^{\frac{t-a}{b}}\right]^{-1}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{b} \left[1 + e^{-\frac{t-a}{b}}\right]^{-1}$	IHR with $\lim_{t \rightarrow \infty} h(t) = \frac{1}{b}$
변수		$t \in \mathbf{R}$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 38: Logistic 분포함수에 기반한 척도 함수

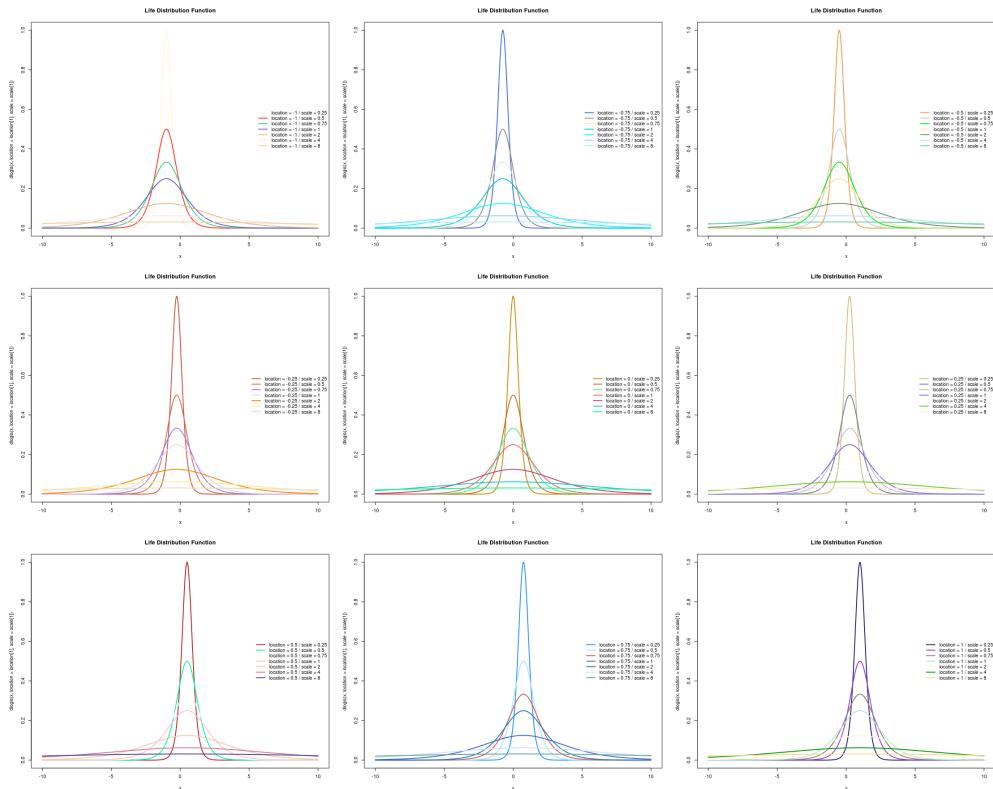


Figure 84: Logistic Distribution에 기반한 수명분포함수

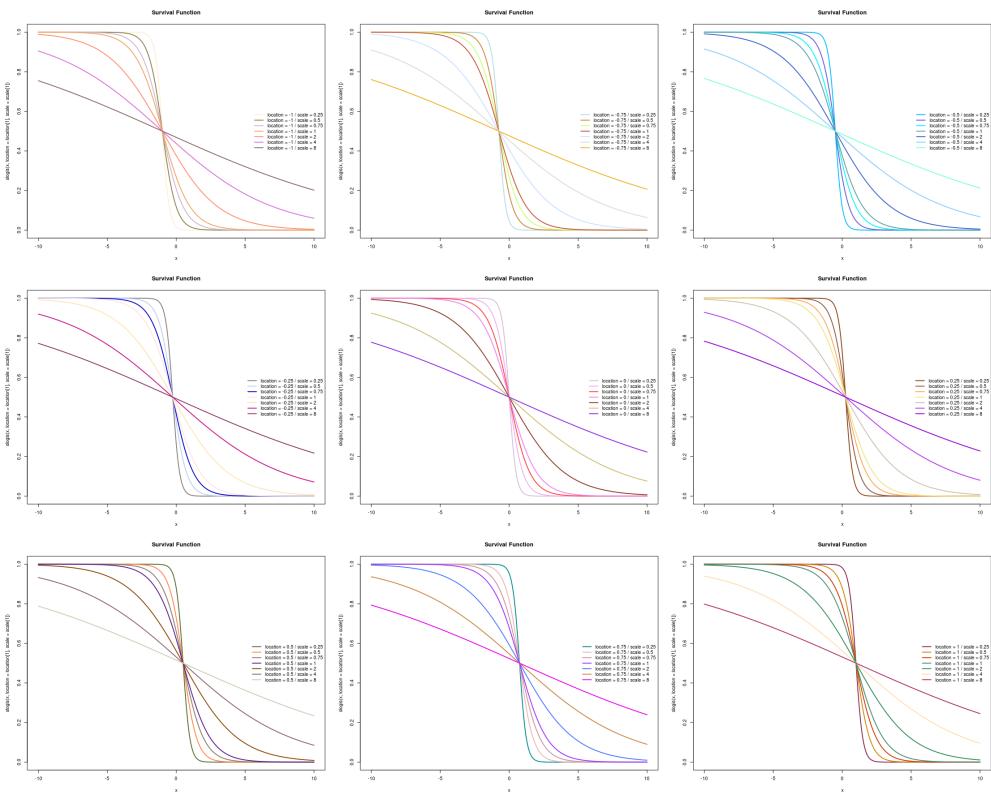


Figure 85: Logistic Distribution에 기반한 생존함수

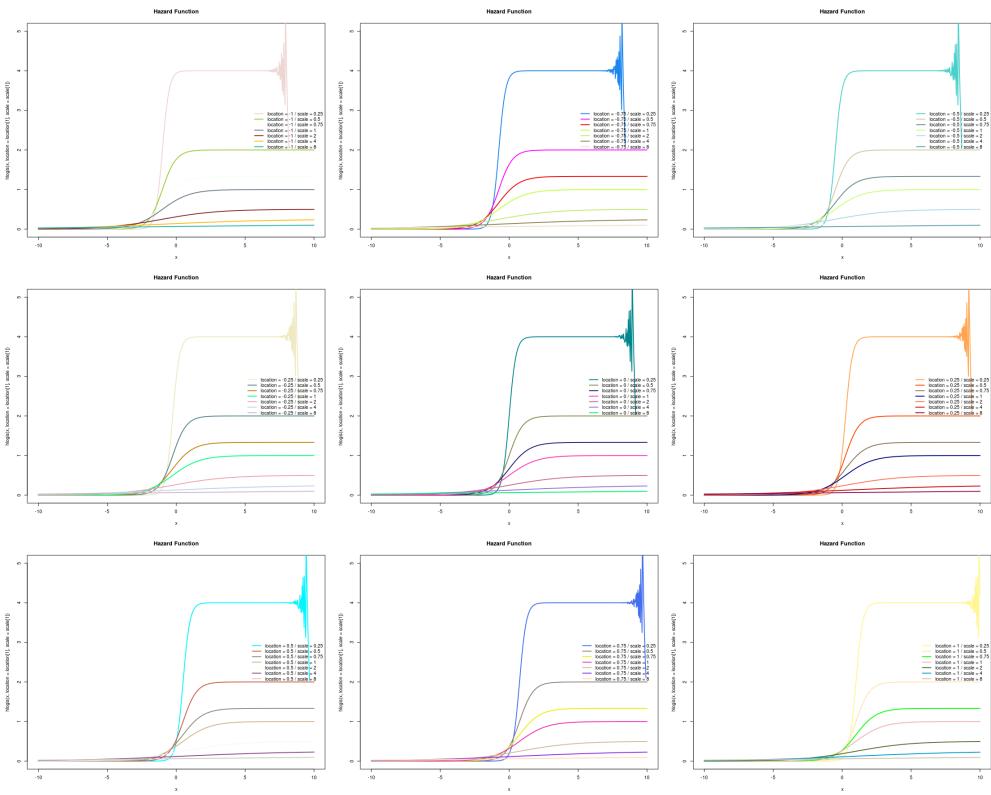


Figure 86: Logistic Distribution에 기반한 위험함수

Code 31. Logistic Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Logistic Distribution
6 ### parameter
7 location = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dlogis(x, location, scale)
16
17
18 ### 분위수 함수
19 qlogis(x, location, scale)
20
21
22 ### 난수 함수
23 rlogis(x, location, scale)
24
25
26 ### 누적분포함수
27 plogis(x, location, scale)
28
29
30 ### 생존함수
31 slogis = function (x, location = 1, scale = 0)
32 {
33   fx = 1 - plogis(x, location, scale)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hlogis = function (x, location = 1, scale = 0)
40 {
41   fx = dlogis(x, location, scale) / slogis(x, location, scale)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.logis_seq = function(x, location = 1, scale = 0, xlim=c(0, 10), ylim=c(0, 5), func="dlogis")
51 {
52   color=colorPalette(300)
53
54   len_location = length(location) # location 파라메터의 길이
55   len_scale = length(scale) # scale 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_location) ### 파라메터: location
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dlogis") # 수명분포
64     {
65       plot(x, dlogis(x, location=location[i], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[i], lwd=2, type = 'n', main="Life Distribution Function")
66       for (j in 1:len_scale) ### 파라메터: scale
67       {
68         lines(x, dlogis(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
71       }
72     }
73     else if (func == "plogis") # 누적분포함수
74     {
75       plot(x, plogis(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
76       for (j in 1:len_scale) ### 파라메터: scale
77       {
78         lines(x, plogis(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
79         color_counter = color_counter + 1;
80         legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
81       }
82     }
83     else if (func == "slogis") # 생존함수
84     {
85       plot(x, slogis(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
86     }
87   }
88 }
```

```

86     for (j in 1:len_scale) ### 파라미터: scale
87     {
88       lines(x, slogis(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
91     }
92   }
93 else if (func == "hlogis") # 위험함수
94 {
95   plot(x, hlogis(x, location=location[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
96   for (j in 1:len_scale) ### 파라미터: scale
97   {
98     lines(x, hlogis(x, location=location[i], scale=scale[j]), col=color[color_counter], lwd=2);
99     color_counter = color_counter + 1;
100    legend_name = c(legend_name, paste("location = ", location[i], " / scale = ", scale[j], sep=""))
101  }
102 }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.logis_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="dlogis")
109
110 par(mfrow = c(3, 3))
111 plot.logis_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="plogis")
112
113 par(mfrow = c(3, 3))
114 plot.logis_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="slogis")
115
116 par(mfrow = c(3, 3))
117 plot.logis_seq(x, location, scale, xlim=c(min(x), max(x)), ylim=c(0, 5), func="hlogis")

```

14.22.1 Log-logistic Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(\frac{t-a}{b} \right)^{c-1} [1 + (\frac{t-a}{b})^c]^{-2}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$[1 + (\frac{t-a}{b})^c]^{-1}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c}{b} \left(\frac{t-a}{b} \right)^{c-1} [1 + (\frac{t-a}{b})^c]^{-1}$	
변수		$t \geq a$	
파라미터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 39: Log-logistic 분포함수에 기반한 척도 함수

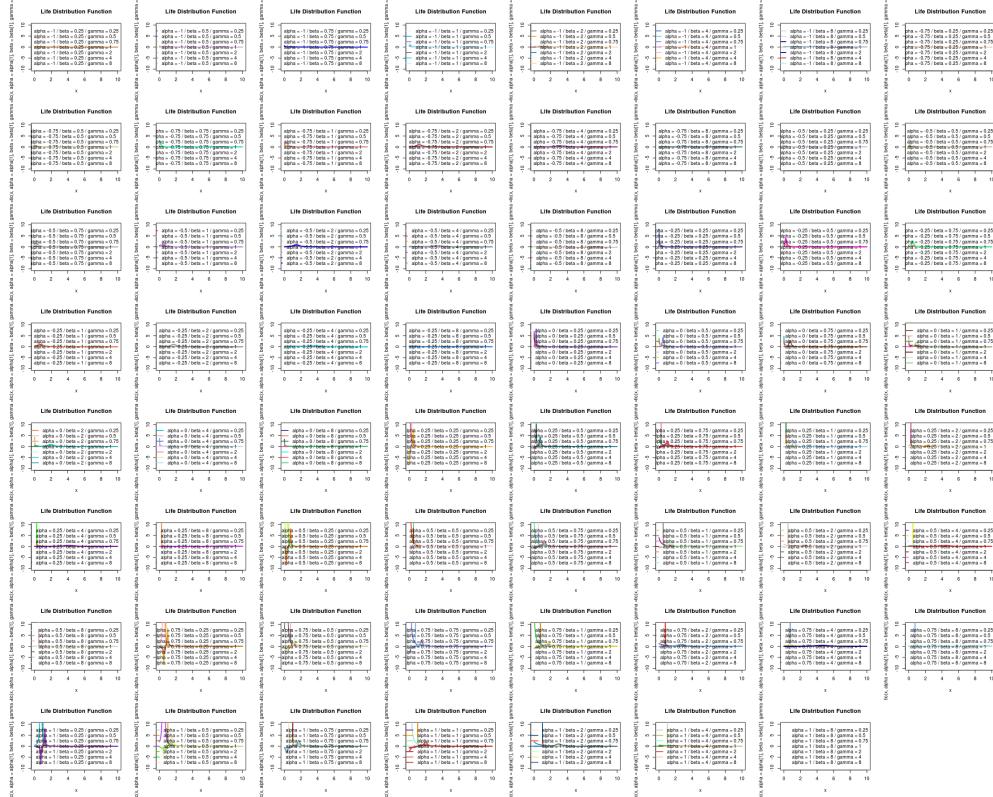


Figure 87: Log-logistic Distribution에 기반한 수명분포함수



Figure 88: Log-logistic Distribution에 기반한 생존함수



Figure 89: Log-logistic Distribution에 기반한 위험함수

Code 32. Log-logistic Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 # https://www.rdocumentation.org/packages/actuar/versions/2.3-1/topics/Loglogistic
2
3 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
4 source("colorPalette.R")
5
6
7 ##### loglogistic Distribution
8 ### parameter
9 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
10 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
11 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
12
13 ### input varialbe
14 x = seq(0, 10, length.out = 1000)
15
16
17 ### 수명 분포
18 dloglogistic = function(x, alpha = 1, beta = 1, gamma = 1)
19 {
20   fx = (gamma/beta) * ((x - alpha)/beta)^(gamma - 1) * (1 + ((x - alpha) / beta)^gamma)^(-2)
21   return(fx)
22 }
23
24
25 ### 누적분포함수
26 ploglogistic = function(x, alpha = 1, beta = 1, gamma = 1)
27 {
28   fx = -(sloglogistic(x, alpha, beta) - 1)
29   return(fx)
30 }
31
32
33 ### 생존함수
34 sloglogistic = function (x, alpha = 1, beta = 1, gamma = 1)
35 {
36   fx = (gamma/beta) * ((x - alpha) / beta)^(gamma - 1) * (1 + ((x - alpha) / beta)^gamma)^(-1)
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hloglogistic = function (x, alpha = 1, beta = 1, gamma = 1)
43 {
44   fx = dloglogistic(x, alpha, beta, gamma) / sloglogistic(x, alpha, beta, gamma)
45   return(fx)
46 }
47
48
49
50
51
52 ##### Plot
53 plot.logistic_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dloglogistic")
54 {
55   color=colorPalette(300)
56
57   len_alpha = length(alpha) # alpha 파라미터의 길이
58   len_beta = length(beta) # beta 파라미터의 길이
59   len_gamma = length(gamma) # gamma 파라미터의 길이
60
61   color_counter = 1
62   for (i in 1:len_alpha) ### 파라미터: alpha
63   {
64     if (func=="dloglogistic") # 수명분포
65     {
66       for (j in 1:len_beta) ### 파라미터: beta
67       {
68         color_counter_init = color_counter
69         legend_name = NULL;
70         plot(x, dloglogistic(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2,
71             type = 'n', main="Life Distribution Function")
72         for (k in 1:len_gamma) ### 파라미터: gamma
73         {
74           lines(x, dloglogistic(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
75           color_counter = color_counter + 1;
76           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
77         }
78         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
79       }
80     }
81   else if (func == "ploglogistic") # 누적분포함수
82   {
83     for (j in 1:len_beta) ### 파라미터: beta
84     {
85       color_counter_init = color_counter
86       legend_name = NULL;
87       plot(x, ploglogistic(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2,
88             type = 'n', main="Survival Function")
89     }
90   }
91 }

```

```

87     type = 'n', main="Cumulative Distribution Function")
88     for (k in 1:len_gamma) ### 파라메터: gamma
89     {
90       lines(x, ploglogistic(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
91       color_counter = color_counter + 1;
92       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
93     }
94   }
95 }
96 else if (func == "sloglogistic") # 생존함수
97 {
98   for (j in 1:len_beta) ### 파라메터: beta
99   {
100     color_counter_init = color_counter
101     legend_name = NULL;
102     plot(x, sloglogistic(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2,
103       type = 'n', main="Survival Function")
104     for (k in 1:len_gamma) ### 파라메터: gamma
105     {
106       lines(x, sloglogistic(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
107       color_counter = color_counter + 1;
108       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
109     }
110   }
111 }
112 else if (func == "hloglogistic") # 위험함수
113 {
114   for (j in 1:len_beta) ### 파라메터: beta
115   {
116     color_counter_init = color_counter
117     legend_name = NULL;
118     plot(x, hloglogistic(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2,
119       type = 'n', main="Hazard Function")
120     for (k in 1:len_gamma) ### 파라메터: gamma
121     {
122       lines(x, hloglogistic(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
123       color_counter = color_counter + 1;
124       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
125     }
126   }
127 }
128 }
129 }
130 par(mfrow = c(8, 8))
131 plot.loglogistic_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dloglogistic")
132
133 par(mfrow = c(8, 8))
134 plot.loglogistic_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="ploglogistic")
135
136 par(mfrow = c(8, 8))
137 plot.loglogistic_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sloglogistic")
138
139 par(mfrow = c(8, 8))
140 plot.loglogistic_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hloglogistic")
141
```

14.22.2 Half-logistic Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{2e^{\frac{t-a}{b}}}{b \left[1 + e^{\frac{t-a}{b}} \right]^2}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$2 \left[1 + e^{\frac{t-a}{b}} \right]^{-1}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\left[b \left(1 + e^{\frac{t-a}{b}} \right) \right]^{-1}$	IHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 40: Half-logistic 분포함수에 기반한 척도 함수

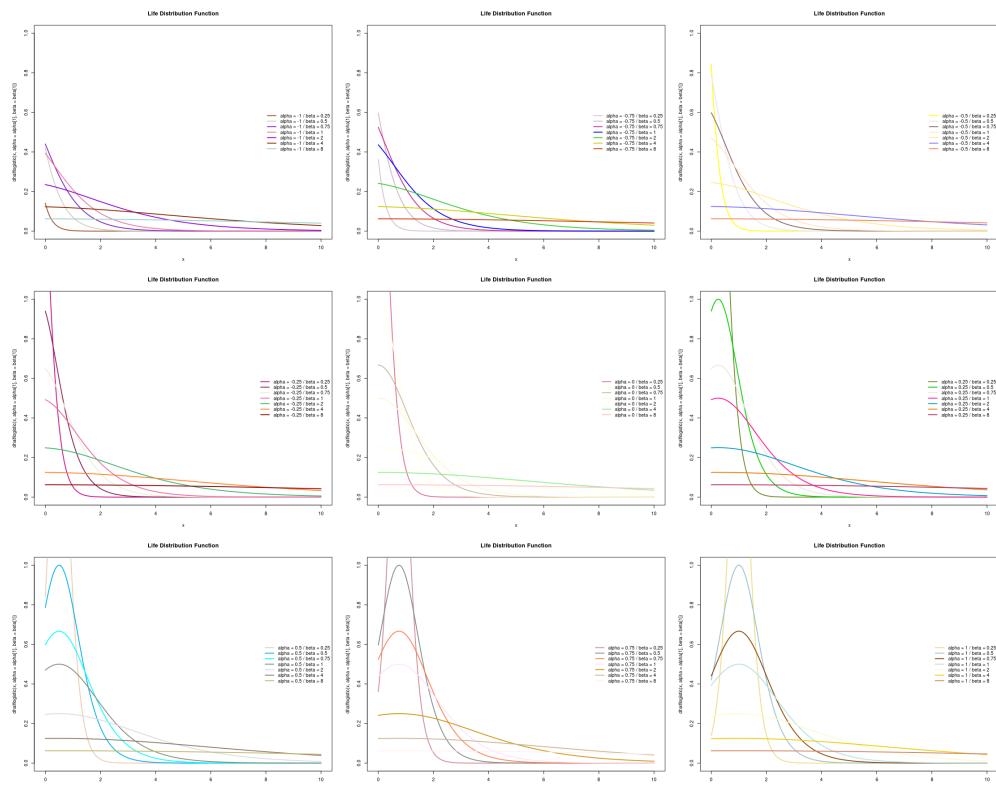


Figure 90: Half-logistic Distribution에 기반한 수명분포함수

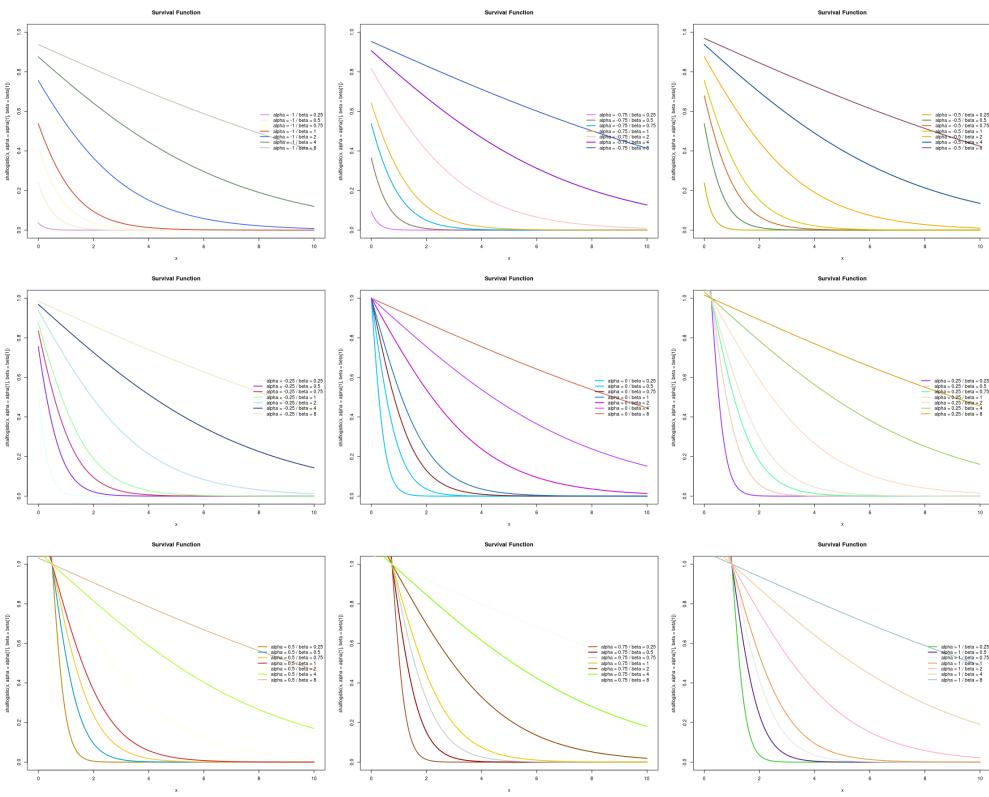


Figure 91: Half-logistic Distribution에 기반한 생존함수

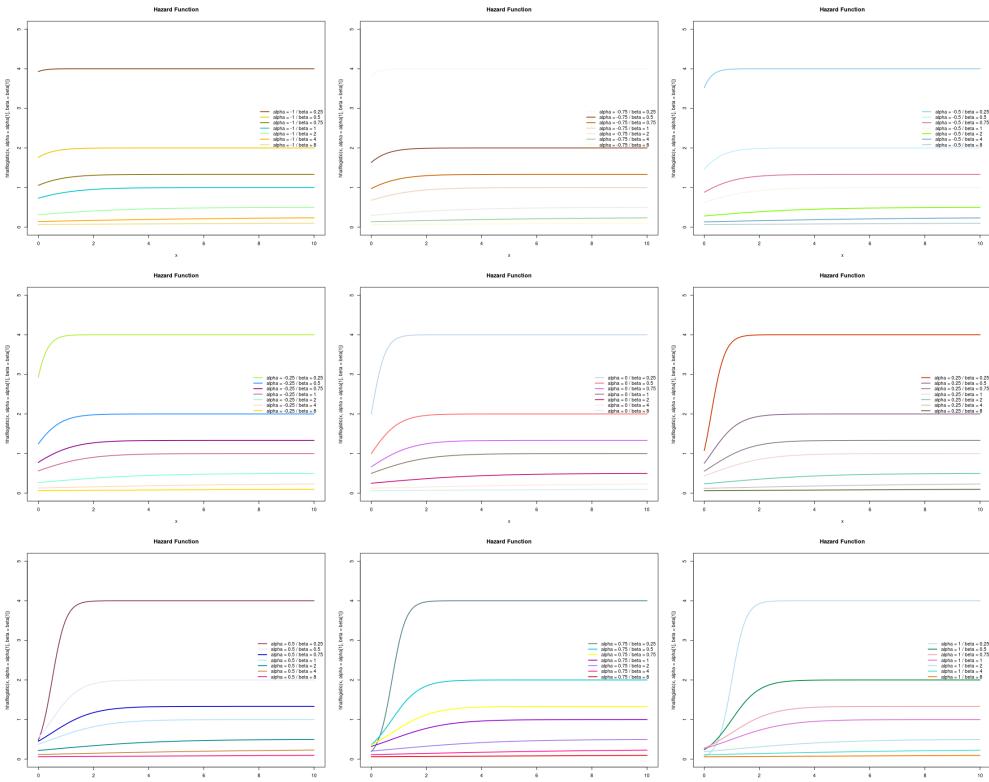


Figure 92: Half-logistic Distribution에 기반한 위험함수

Code 33. Half-logistic Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Half-logistic Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8) # s (beta)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dhalflogistic = function(x, alpha = 0, beta = 1)
16 {
17   fx = (2 * exp((x-alpha)/beta)) / (beta * (1 + exp((x-alpha)/beta))^2)
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 phalflogistic = function(x, alpha = 0, beta = 1)
24 {
25   Fx = -(shalflogistic(x, alpha = alpha, beta = beta) - 1)
26   return(Fx)
27 }
28
29
30 ### 생존함수
31 shalflogistic = function(x, alpha = 0, beta = 1)
32 {
33   fx = 2 * (1 + exp((x-alpha)/beta)) ^ (-1)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hhalflogistic = function(x, alpha = 0, beta = 1)
40 {
41   fx = dhalflogistic(x, alpha = alpha, beta = beta) / shalflogistic(x, alpha = alpha, beta = beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.halflogistic_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dhalflogistic")
51 {
52   color=colorPalette(300)
53
54   len_alpha = length(alpha) # m 파라미터의 길이
55   len_beta = length(beta) # s 파라미터의 길이
56
57   color_counter = 1
58   for (i in 1:len_alpha) ### 파라미터: m
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dhalflogistic") # 수명분포
64     {
65       plot(x, dhalflogistic(x, alpha=alpha[i], beta=beta[i]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
66       for (j in 1:len_beta) ### 파라미터: s
67       {
68         lines(x, dhalflogistic(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
71       }
72     }
73     else if (func == "phalflogistic") # 누적분포함수
74     {
75       plot(x, phalflogistic(x, alpha=alpha[i], beta=beta[i]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
76       for (j in 1:len_beta) ### 파라미터: s
77       {
78         lines(x, phalflogistic(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
79         color_counter = color_counter + 1;
80         legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
81       }
82     }
83     else if (func == "shalflogistic") # 생존함수
84     {
85       plot(x, shalflogistic(x, alpha=alpha[i], beta=beta[i]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
86     }
87   }
88 }
```

```

86     for (j in 1:len_beta) ### 파라메터: s
87     {
88       lines(x, shalflogistic(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
91     }
92   }
93 else if (func == "hhalflogistic") # 위험함수
94 {
95   plot(x, hhalflogistic(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
96         "Hazard Function")
97   for (j in 1:len_beta) ### 파라메터: s
98   {
99     lines(x, hhalflogistic(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("alpha = ", m[i], " / beta = ", s[j], sep=""))
102  }
103  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.halflogistic_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="dhalflogistic")
109
110 par(mfrow = c(3, 3))
111 plot.halflogistic_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="phalflogistic")
112
113 par(mfrow = c(3, 3))
114 plot.halflogistic_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="shalflogistic")
115
116 par(mfrow = c(3, 3))
117 plot.halflogistic_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="hhalflogistic")

```

14.22.3 Generalized Logistic Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{B(c,d)} \frac{e^{-\frac{dt}{b}}}{b \left[1+e^{-\frac{t}{b}}\right]^{c+d}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - I_{\left[1+e^{-\frac{t}{b}}\right]^{-1}}(c, d)$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	IHR with $\lim_{t \rightarrow \infty} h(t) = \frac{d}{b}$
변수		$t \in \mathbf{R}$	
파라미터		$b > 0, c > 0, d > 0$	

Table 41: Generalized logistic 분포함수에 기반한 척도 함수

14.23 Lomax Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(1 + \frac{t-a}{b}\right)^{-(c+1)}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\left(1 + \frac{t-a}{b}\right)^{-c}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c}{t-a-b}$	DHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 42: Lomax 분포함수에 기반한 척도 함수

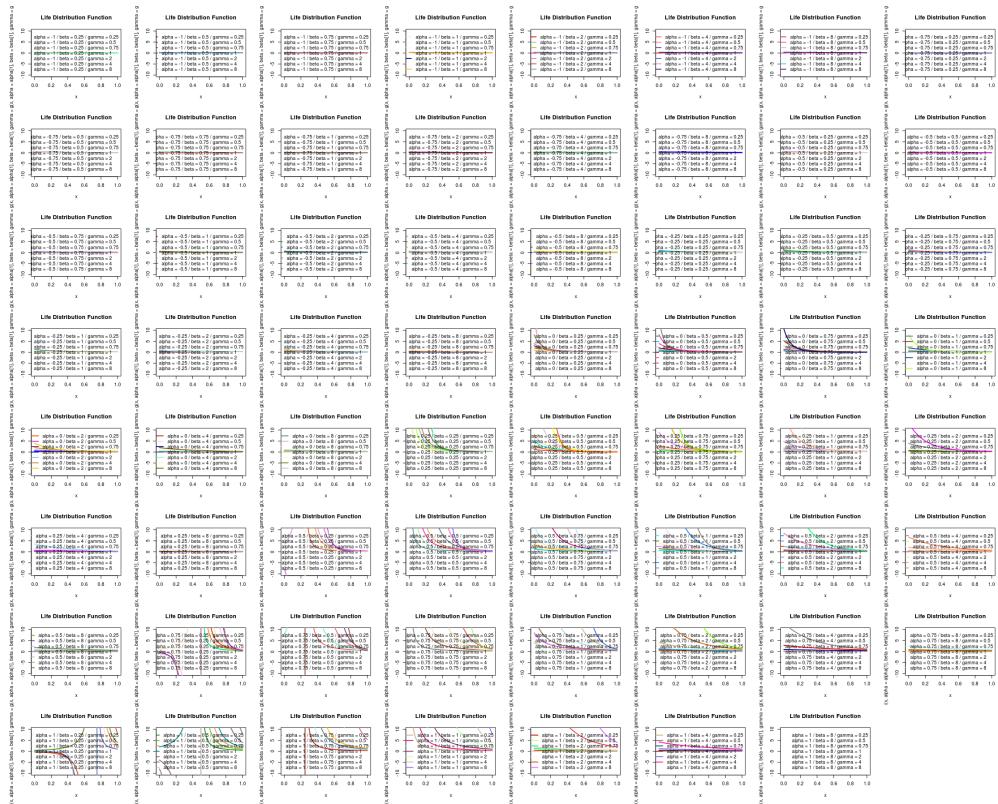


Figure 93: Lomax Distribution에 기반한 수명분포함수



Figure 94: Lomax Distribution에 기반한 생존함수



Figure 95: Lomax Distribution에 기반한 위험함수

Code 34. Lomax Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 # https://www.rdocumentation.org/packages/VGAM/versions/0.7-7/topics/Lomax
2
3 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
4 source("colorPalette.R")
5
6
7 ##### lomax Distribution
8 ### parameter
9 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
10 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
11 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
12
13 ### input varialbe
14 x = seq(0, 1, length.out = 1000)
15
16
17 ### 수명 분포
18 dlomax = function(x, alpha = 1, beta = 1, gamma = 1)
19 {
20   fx = (gamma / beta) * (1 + ((x - alpha)/beta))^{-(gamma+1)}
21   return(fx)
22 }
23
24
25 ### 누적분포함수
26 plomax = function(x, alpha = 1, beta = 1, gamma = 1)
27 {
28   fx = -(slomax(x, alpha, beta) - 1)
29   return(fx)
30 }
31
32
33 ### 생존함수
34 slomax = function (x, alpha = 1, beta = 1, gamma = 1)
35 {
36   fx = (1 + ((x - alpha)/beta))^{-gamma}
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hlomax = function (x, alpha = 1, beta = 1, gamma = 1)
43 {
44   fx = dlomax(x, alpha, beta, gamma) / slomax(x, alpha, beta, gamma)
45   return(fx)
46 }
47
48
49
50
51 ##### Plot
52 plot.lomax_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dlomax")
53 {
54   color=colorPalette(300)
55
56   len_alpha = length(alpha) # alpha 파라미터의 길이
57   len_beta = length(beta) # beta 파라미터의 길이
58   len_gamma = length(gamma) # gamma 파라미터의 길이
59
60   color_counter = 1
61   for (i in 1:len_alpha) ### 파라미터: alpha
62   {
63     if (func=="dlomax") # 수명분포
64     {
65       for (j in 1:len_beta) ### 파라미터: beta
66       {
67         color_counter_init = color_counter
68         legend_name = NULL;
69         plot(x, dlomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
70           'n', main="Life Distribution Function")
71         for (k in 1:len_gamma) ### 파라미터: gamma
72         {
73           lines(x, dlomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
74           color_counter = color_counter + 1;
75           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
76         }
77         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
78       }
79     }
80   else if (func == "plomax") # 누적분포함수
81   {
82     for (j in 1:len_beta) ### 파라미터: beta
83     {
84       color_counter_init = color_counter
85       legend_name = NULL;
86       plot(x, plomax(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =

```

```

87     'n', main="Cumulative Distribution Function")
88     for (k in 1:len_gamma) ### 파라미터: gamma
89     {
90         lines(x, plomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
91         color_counter = color_counter + 1;
92         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
93     }
94     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
95   }
96 else if (func == "slomax") # 생존함수
97 {
98   for (j in 1:len_beta) ### 파라미터: beta
99   {
100     color_counter_init = color_counter
101     legend_name = NULL;
102     plot(x, slomax(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
103       'n', main="Survival Function")
104     for (k in 1:len_gamma) ### 파라미터: gamma
105     {
106         lines(x, slomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
107         color_counter = color_counter + 1;
108         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
109     }
110     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
111   }
112 else if (func == "hlomax") # 위험함수
113 {
114   for (j in 1:len_beta) ### 파라미터: beta
115   {
116     color_counter_init = color_counter
117     legend_name = NULL;
118     plot(x, hlomax(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
119       'n', main="Hazard Function")
120     for (k in 1:len_gamma) ### 파라미터: gamma
121     {
122         lines(x, hlomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
123         color_counter = color_counter + 1;
124         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
125     }
126     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
127   }
128 }
129 }
130 par(mfrow = c(8, 8))
131 plot.lomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dlomax")
132 par(mfrow = c(8, 8))
133 plot.lomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="plomax")
134 par(mfrow = c(8, 8))
135 plot.lomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="slomax")
136 par(mfrow = c(8, 8))
137 plot.lomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="hlomax")
138 par(mfrow = c(8, 8))
139 plot.lomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hlomax")
140
141
```

14.23.1 Generalized Lomax Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} \left[1 + c \frac{t-a}{b} \right]^{-((1+\frac{1}{c}))}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$(1 + c \frac{t-a}{b})^{-\frac{1}{c}}$ for $c \neq 0$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{b} (1 + c \frac{t-a}{b})^{-1}$	IHR for $c < 0$ DHR for $c > 0$
변수		$\begin{cases} a \leq t \leq a - \frac{b}{c} & \text{for } c < 0 \\ t \geq a & \text{for } c > 0 \end{cases}$	
파라메터		$a \in \mathbf{R}, b > 0, c \in \mathbf{R}$	

Table 43: Generalized Lomax 분포함수에 기반한 척도 함수

이 분포는 generalized Pareto distribution of the second kind로도 알려져 있다.

또한 $\lim_{c \rightarrow 0} f(t) = \left(\frac{1}{b} e^{-\frac{t-a}{b}} \right)^{-\frac{1}{c}}$ 가 되며, 이는 exponential distribution이다.



Figure 96: Generalized Lomax Distribution에 기반한 수명분포함수

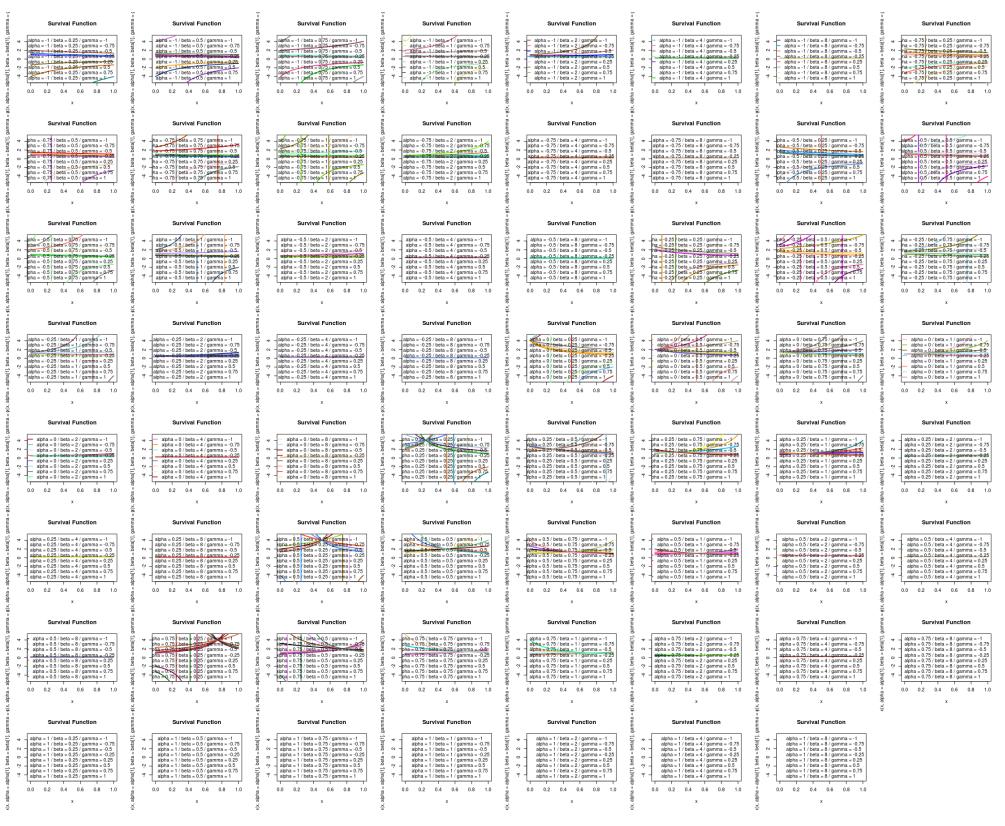


Figure 97: Generalized Lomax Distribution에 기반한 생존함수



Figure 98: Generalized Lomax Distribution에 기반한 위험함수

Code 35. Generalized Lomax Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 # https://www.rdocumentation.org/packages/VGAM/versions/0.7-7/topics/glomax
2
3 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
4 source("colorPalette.R")
5
6
7 ##### glomax Distribution
8 ### parameter
9 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
10 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
11 gamma = c(-1, -0.75, -0.5, -0.25, 0.25, 0.5, 0.75, 1)
12
13 ### input varialbe
14 x = seq(0, 1, length.out = 1000)
15
16
17 ### 수명 분포
18 dglomax = function(x, alpha = 1, beta = 1, gamma = 1)
19 {
20   fx = (1/beta) * (1 + gamma * ((x - alpha)/beta))^{-(1 + 1/gamma)}
21   return(fx)
22 }
23
24
25 ### 누적분포함수
26 pglomax = function(x, alpha = 1, beta = 1, gamma = 1)
27 {
28   fx = -(sglomax(x, alpha, beta) - 1)
29   return(fx)
30 }
31
32
33 ### 생존함수
34 sglomax = function (x, alpha = 1, beta = 1, gamma = 1)
35 {
36   fx = (1/beta) * (1 + gamma * ((x - alpha)/beta))^{(-1)}
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hglomax = function (x, alpha = 1, beta = 1, gamma = 1)
43 {
44   fx = dglomax(x, alpha, beta, gamma) / sglomax(x, alpha, beta, gamma)
45   return(fx)
46 }
47
48
49
50
51
52 ##### Plot
53 plot.glomax_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dglomax")
54 {
55   color=colorPalette(300)
56
57   len_alpha = length(alpha) # alpha 파라미터의 길이
58   len_beta = length(beta) # beta 파라미터의 길이
59   len_gamma = length(gamma) # gamma 파라미터의 길이
60
61   color_counter = 1
62   for (i in 1:len_alpha) ### 파라미터: alpha
63   {
64     if (func=="dglomax") # 수명분포
65     {
66       for (j in 1:len_beta) ### 파라미터: beta
67       {
68         color_counter_init = color_counter
69         legend_name = NULL;
70         plot(x, dglomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[i], lwd=2, type =
71           'n', main="Life Distribution Function")
72         for (k in 1:len_gamma) ### 파라미터: gamma
73         {
74           lines(x, dglomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
75           color_counter = color_counter + 1;
76           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
77         }
78       }
79     }
80   }
81   else if (func == "pglomax") # 누적분포함수
82   {
83     for (j in 1:len_beta) ### 파라미터: beta
84     {
85       color_counter_init = color_counter

```

```

85     legend_name = NULL;
86     plot(x, pglomax(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
87             'n', main="Cumulative Distribution Function")
88     for (k in 1:len_gamma) ### 파라메터: gamma
89     {
90         lines(x, pglomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
91         color_counter = color_counter + 1;
92         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
93     }
94     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
95   }
96 else if (func == "sglomax") # 생존함수
97 {
98   for (j in 1:len_beta) ### 파라메터: beta
99   {
100     color_counter_init = color_counter
101     legend_name = NULL;
102     plot(x, sglomax(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
103           'n', main="Survival Function")
104     for (k in 1:len_gamma) ### 파라메터: gamma
105     {
106         lines(x, sglomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
107         color_counter = color_counter + 1;
108         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
109     }
110     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
111   }
112 else if (func == "hglomax") # 위험함수
113 {
114   for (j in 1:len_beta) ### 파라메터: beta
115   {
116     color_counter_init = color_counter
117     legend_name = NULL;
118     plot(x, hglomax(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
119           'n', main="Hazard Function")
120     for (k in 1:len_gamma) ### 파라메터: gamma
121     {
122         lines(x, hglomax(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
123         color_counter = color_counter + 1;
124         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
125     }
126     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
127   }
128 }
129 }
130 par(mfrow = c(8, 8))
131 plot.glomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dglomax")
132
133 par(mfrow = c(8, 8))
134 plot.glomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pglomax")
135
136 par(mfrow = c(8, 8))
137 plot.glomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sglomax")
138
139 par(mfrow = c(8, 8))
140 plot.glomax_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hglomax")
141

```

14.24 Makeham Distribution(메이크햄 분포)

Table 44: 메이크햄 분포함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$[1 + \theta(1 - e^{-t})] e^{-[t+\theta(t+e^{-t}-1)]}$
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$e^{-[t+\theta(t+e^{-t}-1)]}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$1 + \theta(1 - e^{-t})$
100p 백분위수	t_p	작성중
변수		$t \geq 0$
파라미터		$\theta \geq 0$

메이크햄 분포는 비모수적 검정 방법의 검정력 비교에 자주 이용된다.

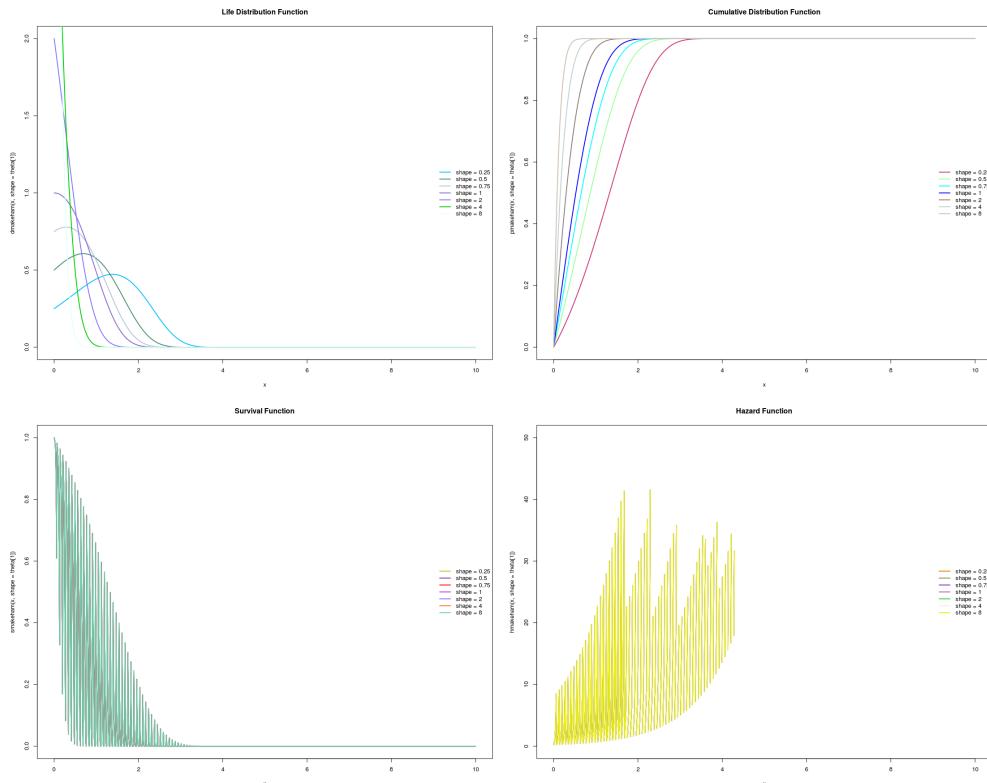


Figure 99: Makeham Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 36. Makeham Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(VGAM)
4
5
6 ##### makeham Distribution
7 ### parameter
8 theta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dmakeham(x, shape = theta)
16
17
18 ### 분위수 함수
19 qmakeham(x, shape = theta)
20
21
22 ### 난수 함수
23 rmakeham(x, shape = theta)
24
25
26 ### 누적분포함수
27 pmakeham(x, shape = theta)
28
29
30 ### 생존함수
31 smakeham = function (x, shape = 1)
32 {
33   fx = 1 - pmakeham(x, shape = theta)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hmakeham = function (x, shape = 1)
40 {
41   fx = dmakeham(x, shape = theta) / smakeham(x, shape = theta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.makeham_seq = function(x, shape = 1, xlim=c(0, 10), ylim=c(0, 5), func="dmakeham")
51 {
52   color=colorPalette(300)
53
54   len_theta = length(theta) # theta 파라메터의 길이
55
56   color_counter = 1
57   color_counter_init = color_counter
58   legend_name = NULL;
59
60
61   if (func=="dmakeham") # 수명분포
62   {
63     plot(x, dmakeham(x, shape=theta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution
Function")
64     for (i in 1:len_theta) ### 파라메터: theta
65     {
66       lines(x, dmakeham(x, shape=theta[i]), col=color[color_counter], lwd=2);
67       color_counter = color_counter + 1;
68       legend_name = c(legend_name, paste("shape = ", theta[i], sep=""))
69     }
70   }
71   else if (func == "pmakeham") # 누적분포함수
72   {
73     plot(x, pmakeham(x, shape=theta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution
Function")
74     for (i in 1:len_theta) ### 파라메터: theta
75     {
76       lines(x, pmakeham(x, shape=theta[i]), col=color[color_counter], lwd=2);
77       color_counter = color_counter + 1;
78       legend_name = c(legend_name, paste("shape = ", theta[i], sep=""))
79     }
80   }
81   else if (func == "smakeham") # 생존함수
82   {
83     plot(x, smakeham(x, shape=theta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
84     for (i in 1:len_theta) ### 파라메터: theta
```

```

85     {
86         lines(x, smakeham(x, shape=theta[i]), col=color[color_counter], lwd=2);
87         color_counter = color_counter + 1;
88         legend_name = c(legend_name, paste("shape = ", theta[i], sep=""))
89     }
90 }
91 else if (func == "hmakeham") # 위험함수
92 {
93     plot(x, hmakeham(x, shape=theta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
94     for (i in 1:len_theta) ### 파라메터: theta
95     {
96         lines(x, hmakeham(x, shape=theta[i]), col=color[color_counter], lwd=2);
97         color_counter = color_counter + 1;
98         legend_name = c(legend_name, paste("shape = ", theta[i], sep=""))
99     }
100 }
101 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
102 }
103
104 par(mfrow = c(2, 2))
105 plot.makeham_seq(x, theta, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dmakeham")
106 plot.makeham_seq(x, theta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pmakeham")
107 plot.makeham_seq(x, theta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="smakeham")
108 plot.makeham_seq(x, theta, xlim=c(min(x), max(x)), ylim=c(0, 50), func="hmakeham")

```

14.25 Maxwell-Boltzmann Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} \sqrt{\frac{2}{\pi}} \left(\frac{t-a}{b}\right)^2 e^{-\frac{1}{2}\left(\frac{t-a}{b}\right)^2}$	
생존함수 (신뢰도함수)	$\begin{aligned} S(t) &= P(T > t) \\ &= 1 - P(T \leq t) \\ &= 1 - F(t) \\ &= e^{-H(t)} \end{aligned}$	$1 - F_{\chi^2_3}\left[\left(\frac{t-a}{b}\right)^2\right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	IHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 45: Maxwell-Boltzmann 분포함수에 기반한 척도 함수

여기서, $F_{\chi^2_3}[\cdot]$ 은 자유도가 3인 χ^2 분포의 CDF

14.26 Muth Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} \left[e^{c(\frac{t-a}{b})} - c \right] e^{-\frac{1}{c}e^{c(\frac{t-a}{b})} + c(\frac{t-a}{b}) + \frac{1}{c}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$e^{-\frac{1}{c}e^{c(\frac{t-a}{b})} + c(\frac{t-a}{b}) + \frac{1}{c}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{b} \left[e^{c(\frac{t-a}{b})} - c \right]$	IHR with $h(a) = \frac{1-c}{b} \forall c$
변수		$t \geq a$	
파라메터		$a \in \mathbb{R}, b > 0, 0 < c \leq 1$	

Table 46: Muth 분포함수에 기반한 척도 함수

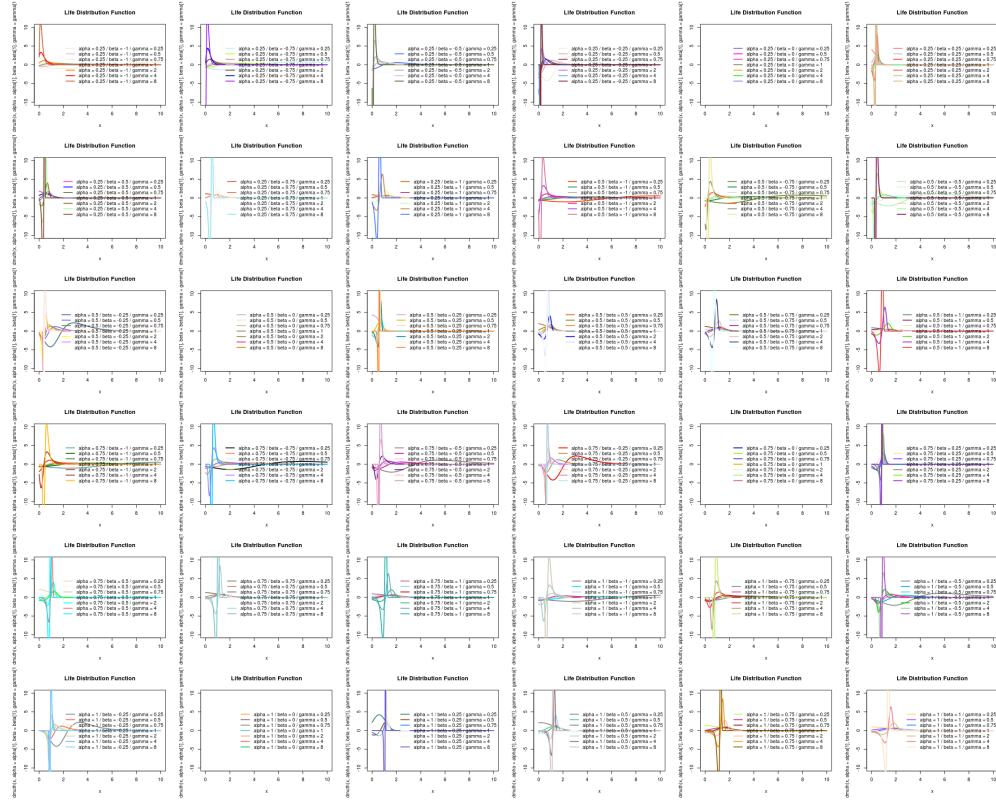


Figure 100: Muth Distribution에 기반한 수명분포함수

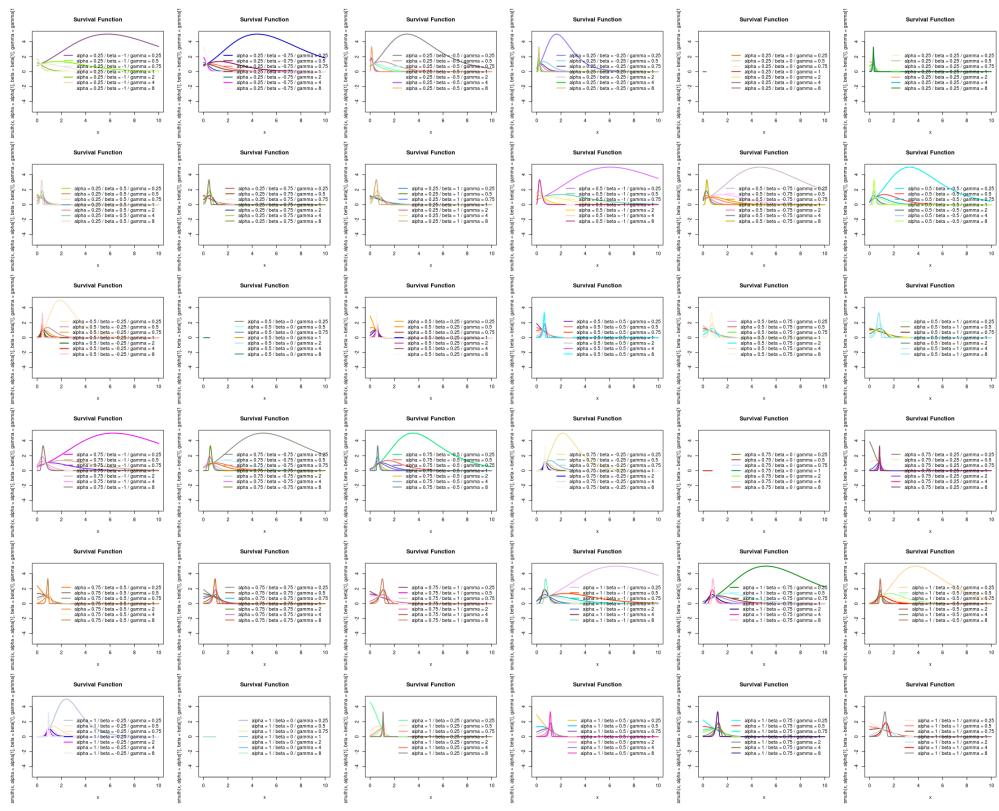


Figure 101: Muth Distribution에 기반한 생존함수

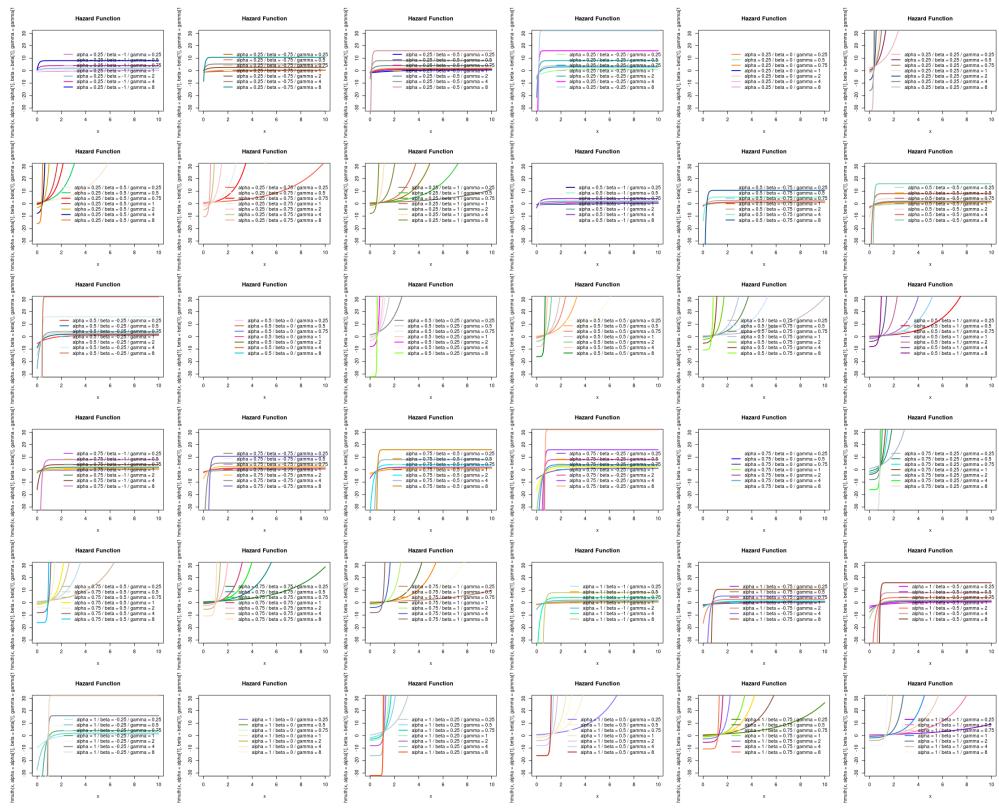


Figure 102: Muth Distribution에 기반한 위험함수

Code 37. Muth Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Gamma Distribution with alpha Parameters
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0.25, 0.5, 0.75, 1)
10
11 ### input varialbe
12 x = seq(0, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dmuth = function(x, alpha=alpha, beta=beta, gamma=gamma)
17 {
18   temp = gamma * ((x - alpha)/beta)
19   fx = (1/beta) * (exp(temp) - gamma) * exp(-(1/gamma)) * exp(temp) + temp + (1/gamma))
20   return(fx)
21 }
22
23
24 ### 누적분포함수
25 pmuth = function(x, alpha=alpha, beta=beta, gamma=gamma)
26 {
27   fx = -(smuth(x, alpha=alpha, beta=beta, gamma=gamma) - 1)
28   return(fx)
29 }
30
31
32 ### 생존함수
33 smuth = function(x, beta=1, gamma=1, alpha=0)
34 {
35   temp = gamma * ((x - alpha)/beta)
36   fx = exp(-(1/gamma)) * exp(temp) + temp + (1/gamma))
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hmuth = function (x, beta=beta, gamma=gamma, alpha=0)
43 {
44   fx = dmuth(x, alpha=alpha, beta=beta, gamma=gamma) / smuth(x, alpha=alpha, beta=beta, gamma=gamma)
45   return(fx)
46 }
47
48
49
50
51
52 ##### Plot
53 plot.muth_seq = function(x, beta = 1, gamma = 1, alpha = 1, xlim=c(0, 10), ylim=c(0, 5), func="dmuth")
54 {
55   color=colorPalette(300)
56
57   len_alpha = length(alpha) # alpha 파라미터의 길이
58   len_beta = length(beta) # beta 파라미터의 길이
59   len_gamma = length(gamma) # gamma 파라미터의 길이
60
61   color_counter = 1
62   for (i in 1:len_alpha) ### 파라미터: alpha
63   {
64     if (func=="dmuth") # 수명분포
65     {
66       for (j in 1:len_beta) ### 파라미터: beta
67       {
68         color_counter_init = color_counter
69         legend_name = NULL;
70         plot(x, dmuth(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
71           'n', main="Life Distribution Function")
72         for (k in 1:len_gamma) ### 파라미터: gamma
73         {
74           lines(x, dmuth(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
75           color_counter = color_counter + 1;
76           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
77         }
78         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
79       }
80     }
81   else if (func == "pmuth") # 누적분포함수
82   {
83     for (j in 1:len_beta) ### 파라미터: beta
84     {
85       color_counter_init = color_counter
86       legend_name = NULL;
87       plot(x, pmuth(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =

```

```

87     n', main="Cumulative Distribution Function")
88     for (k in 1:len_gamma) ### 파라메터: gamma
89     {
90       lines(x, pmuth(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
91       color_counter = color_counter + 1;
92       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
93     }
94   }
95 }
96 else if (func == "smuth") # 생존함수
97 {
98   for (j in 1:len_beta) ### 파라메터: beta
99   {
100     color_counter_init = color_counter
101     legend_name = NULL;
102     plot(x, smuth(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
103           'n', main="Survival Function")
104     for (k in 1:len_gamma) ### 파라메터: gamma
105     {
106       lines(x, smuth(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
107       color_counter = color_counter + 1;
108       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
109     }
110   }
111 }
112 else if (func == "hmuth") # 위험함수
113 {
114   for (j in 1:len_beta) ### 파라메터: beta
115   {
116     color_counter_init = color_counter
117     legend_name = NULL;
118     plot(x, hmuth(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
119           'n', main="Hazard Function")
120     for (k in 1:len_gamma) ### 파라메터: gamma
121     {
122       lines(x, hmuth(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
123       color_counter = color_counter + 1;
124       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
125     }
126   }
127 }
128 }
129 }
130 par(mfrow = c(6, 6))
131 plot.muth_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dmuth")
132
133 par(mfrow = c(6, 6))
134 plot.muth_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pmuth")
135
136 par(mfrow = c(6, 6))
137 plot.muth_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="smuth")
138
139 par(mfrow = c(6, 6))
140 plot.muth_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hmuth")
141

```

14.27 Normal(Gaussian) Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{\sqrt{2\pi}b} e^{-\frac{(t-a)^2}{2b^2}} = \frac{1}{b}\phi\left(\frac{t-a}{b}\right)$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - \Phi\left(\frac{t-a}{b}\right) = \Phi\left(\frac{a-t}{b}\right)$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{b}\phi\left(\frac{t-a}{b}\right) [\Phi\left(\frac{a-t}{b}\right)]^{-1}$	IHR
변수		$t \in \mathbf{R}$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 47: Normal 분포함수에 기반한 척도 함수

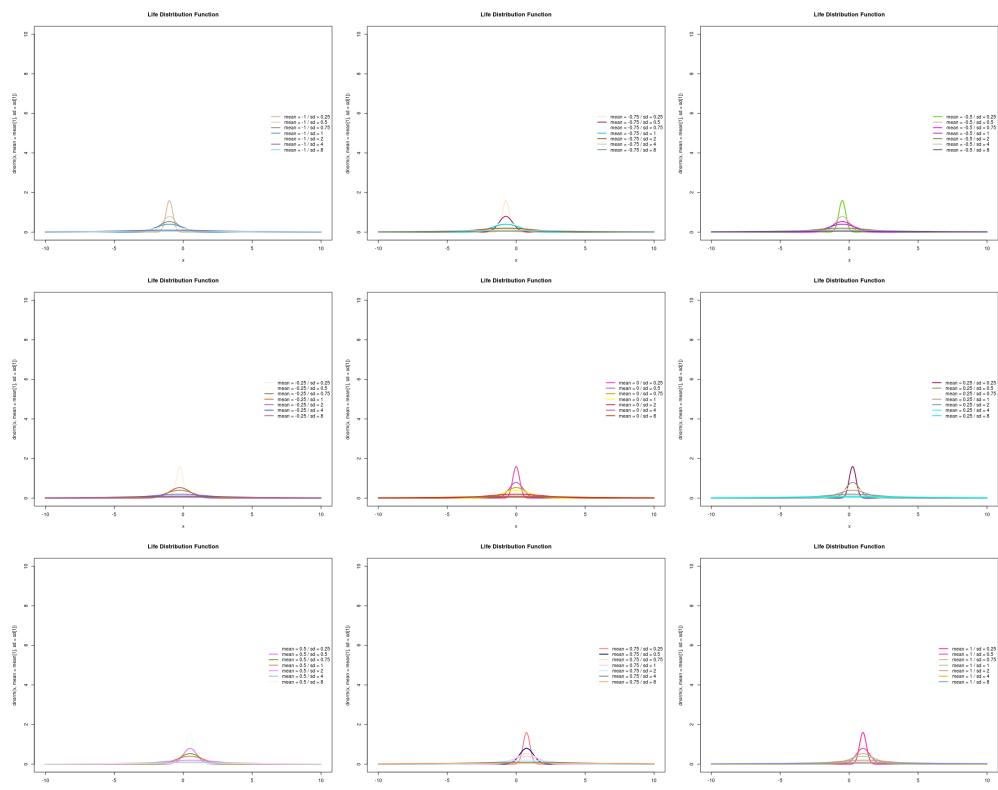


Figure 103: Normal Distribution에 기반한 수명분포함수

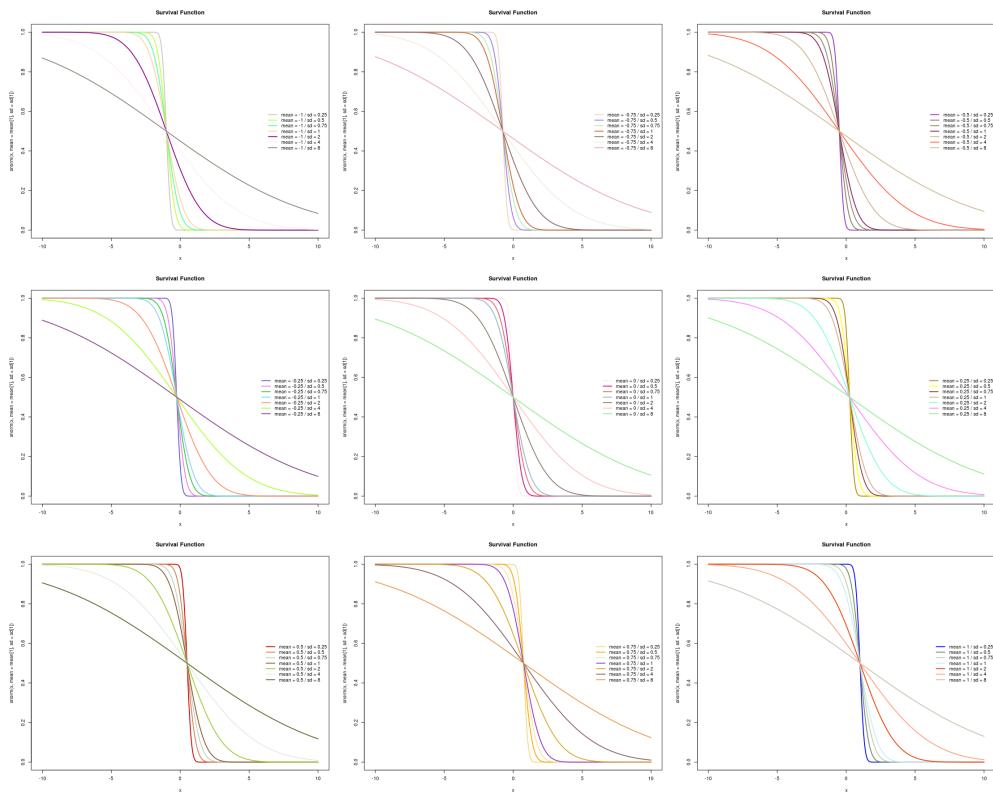


Figure 104: Normal Distribution에 기반한 생존함수

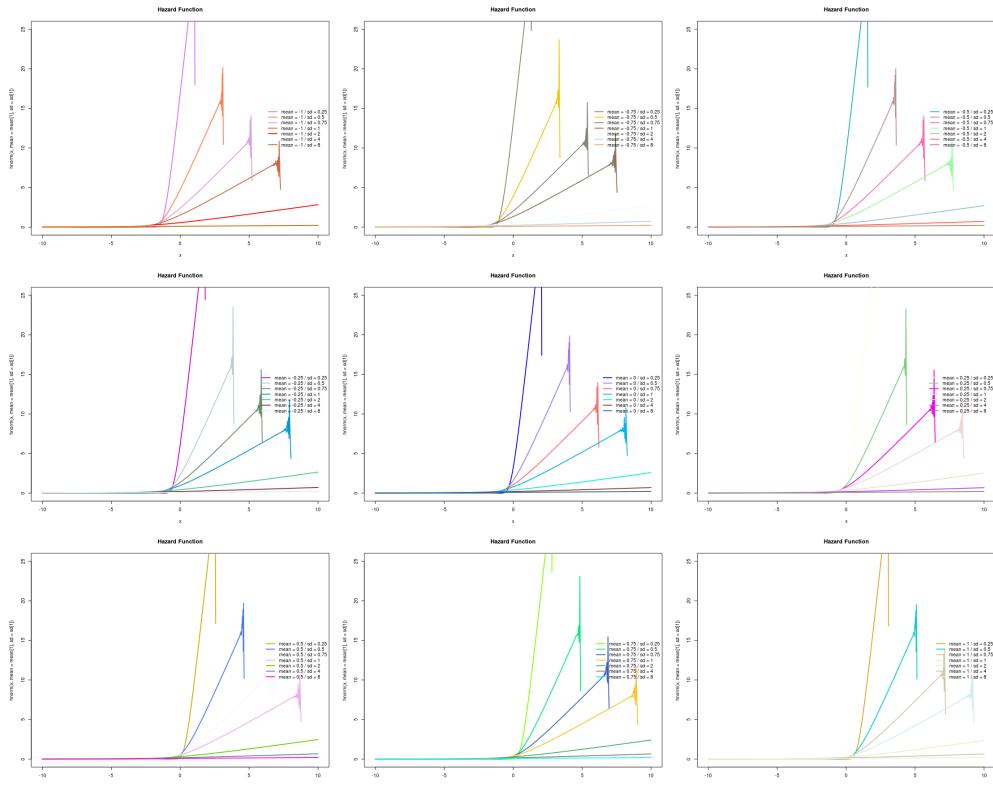


Figure 105: Normal Distribution에 기반한 위험함수

Code 38. Normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Chi-square Distribution
6 ### parameter
7 mean = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 sd = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dnorm(x, mean = mean, sd = sd)
16
17
18 ### 분위수 함수
19 qnorm(x, mean = mean, sd = sd)
20
21
22 ### 난수 함수
23 rnorm(x, mean = mean, sd = sd)
24
25
26 ### 누적분포함수
27 pnorm(x, mean = mean, sd = sd)
28
29
30 ### 생존함수
31 snorm = function (x, mean = 0, sd = 1)
32 {
33   fx = 1 - pnorm(x, mean = mean, sd = sd)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hnorm = function (x, mean = 0, sd = 1)
40 {
41   fx = dnorm(x, mean = mean, sd = sd) / snorm(x, mean = mean, sd = sd)
42   return(fx)
43 }
44
45
46
47
48 ###### Plot
49 plot.norm_seq = function(x, mean = 1, sd = 0, xlim=c(0, 10), ylim=c(0, 5), func="dnorm")
50 {
51   color=colorPalette(300)
52
53   len_mean = length(mean) # mean 파라미터의 길이
54   len_sd = length(sd) # sd 파라미터의 길이
55
56   color_counter = 1
57   for (i in 1:len_mean) ### 파라미터: mean
58   {
59     color_counter_init = color_counter
60     legend_name = NULL;
61
62     if (func=="dnorm") # 수명분포
63     {
64       plot(x, dnorm(x, mean=mean[i], sd=sd[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
65           Distribution Function")
66       for (j in 1:len_sd) ### 파라미터: sd
67       {
68         lines(x, dnorm(x, mean=mean[i], sd=sd[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("mean = ", mean[i], " / sd = ", sd[j], sep=""))
71       }
72     }
73     else if (func == "pnorm") # 누적분포함수
74     {
75       plot(x, pnorm(x, mean=mean[1], sd=sd[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative
76           Distribution Function")
77       for (j in 1:len_sd) ### 파라미터: sd
78       {
79         lines(x, pnorm(x, mean=mean[i], sd=sd[j]), col=color[color_counter], lwd=2);
80         color_counter = color_counter + 1;
81         legend_name = c(legend_name, paste("mean = ", mean[i], " / sd = ", sd[j], sep=""))
82       }
83     }
84     else if (func == "snorm") # 생존함수
85     {
86       plot(x, snorm(x, mean=mean[1], sd=sd[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival
87           Function")
88     }
89   }
90 }
```

```

86     for (j in 1:len_sd) ### 파라미터: sd
87     {
88       lines(x, snorm(x, mean=mean[i], sd=sd[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("mean = ", mean[i], " / sd = ", sd[j], sep=""))
91     }
92   }
93 else if (func == "hnorm") # 위험함수
94 {
95   plot(x, hnorm(x, mean=mean[1], sd=sd[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
96   )
97   for (j in 1:len_sd) ### 파라미터: sd
98   {
99     lines(x, hnorm(x, mean=mean[i], sd=sd[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("mean = ", mean[i], " / sd = ", sd[j], sep=""))
102  }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.norm_seq(x, mean, sd, xlim=c(min(x), max(x)), ylim=c(0, 10), func="dnorm")
109
110 par(mfrow = c(3, 3))
111 plot.norm_seq(x, mean, sd, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pnorm")
112
113 par(mfrow = c(3, 3))
114 plot.norm_seq(x, mean, sd, xlim=c(min(x), max(x)), ylim=c(0, 1), func="snorm")
115
116 par(mfrow = c(3, 3))
117 plot.norm_seq(x, mean, sd, xlim=c(min(x), max(x)), ylim=c(0, 25), func="hnorm")

```

14.27.1 Log-normal Distribution(대수 정규 분포, 로그 정규 분포)

Table 48: 로그정규분포함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$\psi\left(\frac{\ln t - \mu}{\sigma}\right) = \frac{1}{t\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln t - \mu}{\sigma}\right)^2}$
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$1 - \Phi\left(\frac{\ln t - \mu}{\sigma}\right)$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{\psi\left(\frac{\ln t - \mu}{\sigma}\right)}{t\sigma[1 - \Phi\left(\frac{\ln t - \mu}{\sigma}\right)]}$
변수		$t \geq 0$
파라미터	μ (위치 모수; location parameter), σ (척도 모수; scale parameter)	

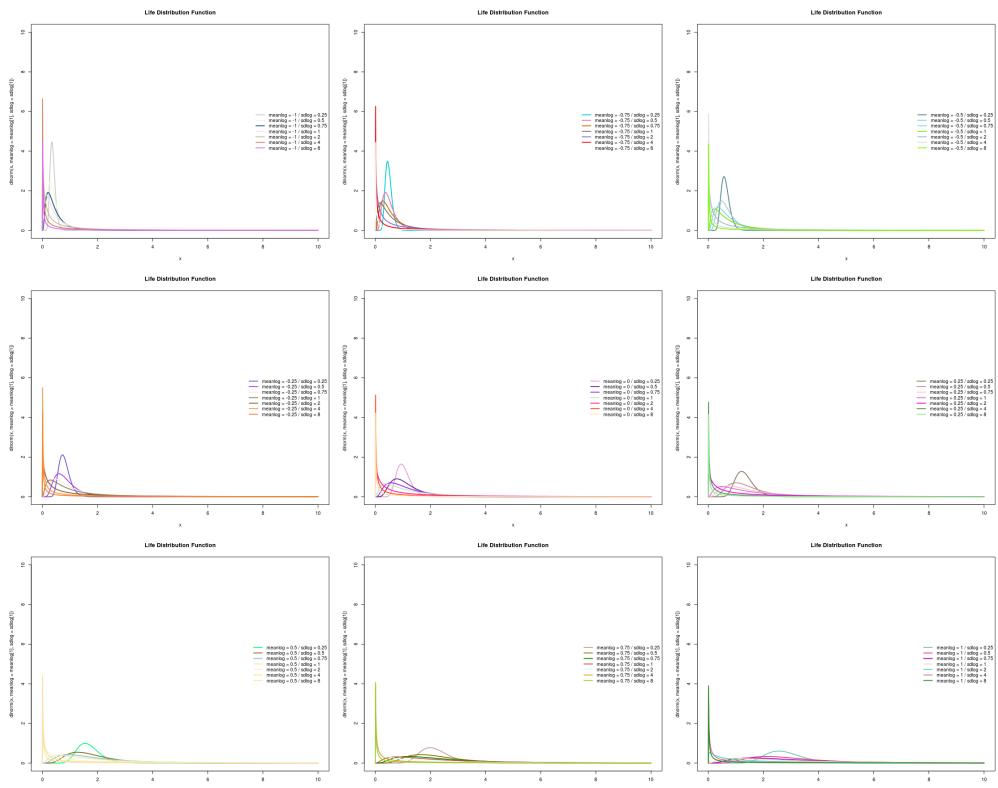


Figure 106: Log-normal Distribution에 기반한 수명분포함수

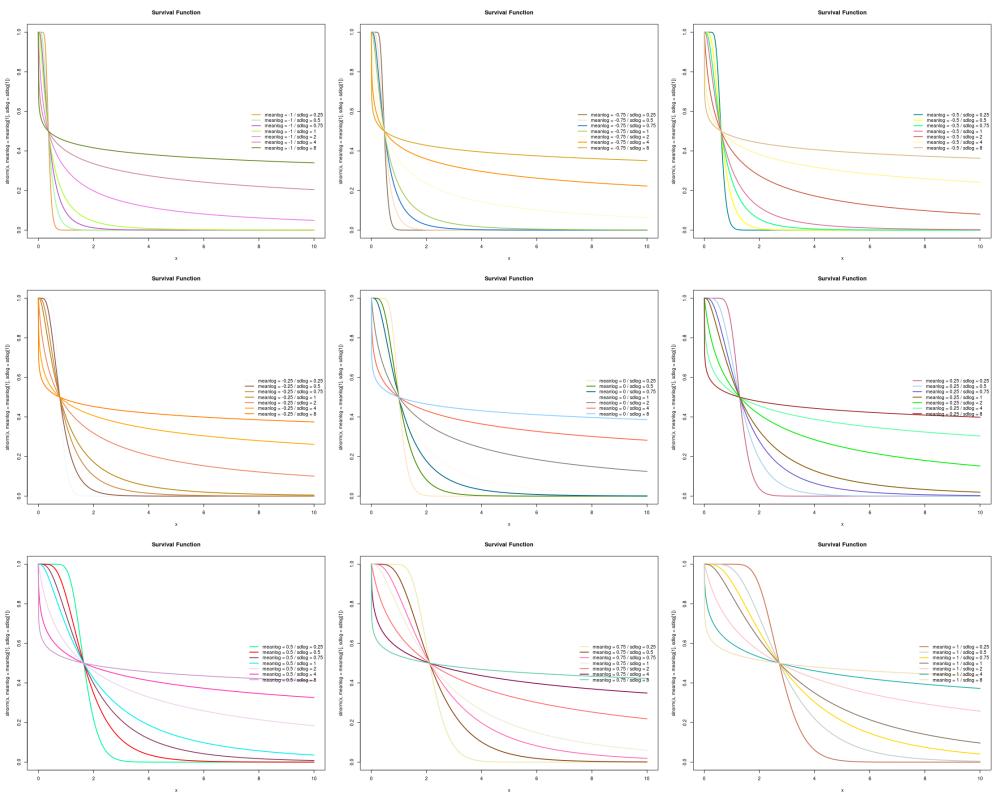


Figure 107: Log-normal Distribution에 기반한 생존함수

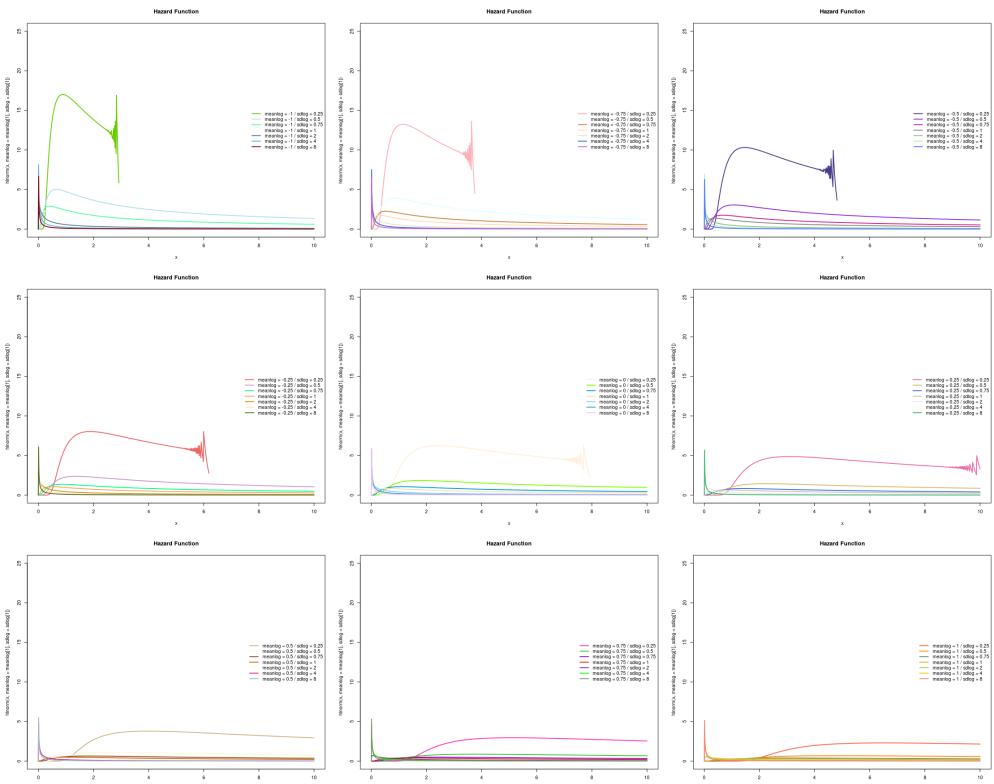


Figure 108: Log-normal Distribution에 기반한 위험함수

Code 39. Log-normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Chi-square Distribution
6 ### parameter
7 meanlog = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 sdlog = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dlnorm(x, meanlog = meanlog, sdlog = sdlog)
16
17
18 ### 분위수 함수
19 qlnorm(x, meanlog = meanlog, sdlog = sdlog)
20
21
22 ### 난수 함수
23 rlnorm(x, meanlog = meanlog, sdlog = sdlog)
24
25
26 ### 누적분포함수
27 plnorm(x, meanlog = meanlog, sdlog = sdlog)
28
29
30 ### 생존함수
31 slnrm = function (x, meanlog = 0, sdlog = 1)
32 {
33   fx = 1 - plnorm(x, meanlog = meanlog, sdlog = sdlog)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hlnorm = function (x, meanlog = 0, sdlog = 1)
40 {
41   fx = dlnorm(x, meanlog = meanlog, sdlog = sdlog) / slnrm(x, meanlog = meanlog, sdlog = sdlog)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.lnorm_seq = function(x, meanlog = 1, sdlog = 0, xlim=c(0, 10), ylim=c(0, 5), func="dlnorm")
51 {
52   color=colorPalette(300)
53
54   len_meanlog = length(meanlog) # meanlog 파라메터의 길이
55   len_sdlog = length(sdlog) # sdlog 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_meanlog) ### 파라메터: meanlog
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dlnorm") # 수명분포
64     {
65       plot(x, dlnorm(x, meanlog=meanlog[i], sdlog=sdlog[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
66       for (j in 1:len_sdlog) ### 파라메터: sdlog
67       {
68         lines(x, dlnorm(x, meanlog=meanlog[i], sdlog=sdlog[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("meanlog = ", meanlog[i], " / sdlog = ", sdlog[j], sep=""))
71       }
72     }
73     else if (func == "plnorm") # 누적분포함수
74     {
75       plot(x, plnorm(x, meanlog=meanlog[1], sdlog=sdlog[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
76       for (j in 1:len_sdlog) ### 파라메터: sdlog
77       {
78         lines(x, plnorm(x, meanlog=meanlog[i], sdlog=sdlog[j]), col=color[color_counter], lwd=2);
79         color_counter = color_counter + 1;
80         legend_name = c(legend_name, paste("meanlog = ", meanlog[i], " / sdlog = ", sdlog[j], sep=""))
81       }
82     }
83     else if (func == "slnrm") # 생존함수
84     {
85       plot(x, slnrm(x, meanlog=meanlog[1], sdlog=sdlog[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
86     }
87   }
88 }
```

```

86     for (j in 1:len_sdlog) ### 파라메터: sdlog
87     {
88       lines(x, slnorm(x, meanlog=meanlog[i], sdlog=sdlog[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("meanlog = ", meanlog[i], " / sdlog = ", sdlog[j], sep=""))
91     }
92   }
93   else if (func == "hlnorm") # 위험함수
94   {
95     plot(x, hlnorm(x, meanlog=meanlog[1], sdlog=sdlog[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=""
96           Hazard Function")
97     for (j in 1:len_sdlog) ### 파라메터: sdlog
98     {
99       lines(x, hlnorm(x, meanlog=meanlog[i], sdlog=sdlog[j]), col=color[color_counter], lwd=2);
100      color_counter = color_counter + 1;
101      legend_name = c(legend_name, paste("meanlog = ", meanlog[i], " / sdlog = ", sdlog[j], sep=""))
102    }
103    legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104  }
105}
106
107 par(mfrow = c(3, 3))
108 plot.lnorm_seq(x, meanlog, sdlog, xlim=c(min(x), max(x)), ylim=c(0, 10), func="dlnorm")
109
110 par(mfrow = c(3, 3))
111 plot.lnorm_seq(x, meanlog, sdlog, xlim=c(min(x), max(x)), ylim=c(0, 1), func="plnorm")
112
113 par(mfrow = c(3, 3))
114 plot.lnorm_seq(x, meanlog, sdlog, xlim=c(min(x), max(x)), ylim=c(0, 1), func="slnorm")
115
116 par(mfrow = c(3, 3))
117 plot.lnorm_seq(x, meanlog, sdlog, xlim=c(min(x), max(x)), ylim=c(0, 25), func="hlnorm")

```

로그정규분포는 고장(재발)시간을 모형화하는데 자주 사용하는 분포이다.

로그정규분포는 정규분포와 밀접한 관련이 있는데, 만일 $T \sim N()$ 을 따른다면, $\ln T \sim N()$ 이 된다.

로그정규분포는 척도 모수 σ 가 커지면, 확률밀도함수의 그래프가 왼쪽으로 치우치는 경향이 있고, 척도 모수 σ 가 작아지면, 확률밀도함수의 그래프가 종 모양의 그래프가 되어간다.

이는, 척도 모수 σ 의 값이 크면, ”초기에” 고장(재발) 발생률이 높다는 것을 의미한다.

로그정규분포의 응용 분야는 다음과 같다.

- 동일하고 독립적인 분포를 하는 양적인 확률변수들의 곱으로 표현되는 확률변수를 나타낸다.
- 여러 개의 요인이 합의 형태가 아닌 곱이나 비례의 형태로 열화과정이 진행되는 경우를 모형화하는 데 적합하다.
- 금속의 피로(fatigue), 반도체나 다이오드와 같은 부속이나 전기절연체의 수명자료 분석에 광범위하게 사용된다.
- 모집단에서 작은 부분을 차지하는 불량품에 의해 고장률(위험률)이 감소하는 부품의 고장 시간을 모형화하는 데 적합하다.

14.27.2 Log-normal Distribution with Lower Threshold

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$e^{-\frac{[\ln(t-a)-\alpha]^2}{2\beta^2}} [\beta(t-a)\sqrt{2\pi}]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\Phi\left[-\frac{\ln(t-a)-\alpha}{\beta}\right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\phi\left[-\frac{\ln(t-a)-\alpha}{\beta}\right] \left[\beta(t-a)\Phi\left(-\frac{\ln(t-a)-\alpha}{\beta}\right)\right]^{-1}$	IDHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, \alpha \in \mathbf{R}, \beta > 0$	

Table 49: Log-normal with lower threshold 분포함수에 기반한 척도 함수

14.27.3 Log-normal Distribution with Upper Threshold

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$e^{-\frac{[\ln(a-t)-\alpha]^2}{2\beta^2}} [\beta(a-t)\sqrt{2\pi}]^{-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\Phi\left[-\frac{\ln(a-t)-\alpha}{\beta}\right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\phi\left[-\frac{\ln(a-t)-\alpha}{\beta}\right] \left[\beta(a-t)\Phi\left(-\frac{\ln(a-t)-\alpha}{\beta}\right)\right]^{-1}$	IHR
변수		$t < a$	
파라미터		$a \in \mathbf{R}, \alpha \in \mathbf{R}, \beta > 0$	

Table 50: Log-normal with upper threshold 분포함수에 기반한 척도 함수

14.27.4 Inverse Normal(Gaussian) Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\sqrt{\frac{b}{2\pi t^3}} e^{-\frac{b(t-a)^2}{2a^2 t}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\Phi\left[\sqrt{\frac{b}{t}}\left(1 - \frac{t}{a}\right)\right] - e^{\frac{2b}{a}} \Phi\left[-\sqrt{\frac{b}{t}}\left(\frac{t}{a} + 1\right)\right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	IDHR
변수		$t > 0$	
파라메터		$a > 0, b > 0$	

Table 51: Inverse-normal 분포함수에 기반한 척도 함수

이 분포는 Wald distribution이라고도 알려져 있다.

Code 40. Inverse Normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(rmtil)
4
5 ##### Chi-square Distribution
6 #### parameter
7 m = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8 s = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 #### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 #### 수명 분포
15 dinvgauss(x, m = 1, s = 1)
16
17
18 #### 분위수 함수
19 qinvgauss(x, m = 1, s = 1)
20
21
22 #### 난수 함수
23 rinvgauss(x, m = 1, s = 1)
24
25
26 #### 누적분포함수
27 pinvgauss(x, m = 1, s = 1)
28
29
30 #### 생존함수
31 sinvgauss = function (x, m = 1, s = 1)
32 {
33   fx = 1 - pinvgauss(x, m = m, s = s)
34   return(fx)
35 }
36
37
38 #### 위험함수
39 hinvgauss = function (x, m = 1, s = 1)
40 {
41   fx = dinvgauss(x, m = m, s = s) / sinvgauss(x, m = m, s = s)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.invgauss_seq = function(x, m = 1, s = 1, xlim=c(0, 10), ylim=c(0, 5), func="dinvgauss")
51 {
52   color=colorPalette(300)
53
54   len_m = length(m) # m 파라미터의 길이
55   len_s = length(s) # s 파라미터의 길이
56
57   color_counter = 1
58   for (i in 1:len_m) ### 파라미터: m
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dinvgauss") # 수명분포
64     {
65       plot(x, dinvgauss(x, m=m[i], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
66       for (j in 1:len_s) ### 파라미터: s
67       {
68         lines(x, dinvgauss(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
69         color_counter = color_counter + 1;
70         legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
71       }
72     }
73   }
74   else if (func == "pinvgauss") # 누적분포함수
75   {
76     plot(x, pinvgauss(x, m=m[1], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
77     for (j in 1:len_s) ### 파라미터: s
78     {
79       lines(x, pinvgauss(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
80       color_counter = color_counter + 1;
81       legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
82     }
83   }
84   else if (func == "sinvgauss") # 생존함수
```

```

85 {
86   plot(x, sinvgauss(x, m=m[1], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
87   for (j in 1:len_s) ### 파라미터: s
88   {
89     lines(x, sinvgauss(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
90     color_counter = color_counter + 1;
91     legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
92   }
93 }
94 else if (func == "hinvgauss") # 위험함수
95 {
96   plot(x, hinvgauss(x, m=m[1], s=s[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
97   for (j in 1:len_s) ### 파라미터: s
98   {
99     lines(x, hinvgauss(x, m=m[i], s=s[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("m = ", m[i], " / s = ", s[j], sep=""))
102  }
103 }
104 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
105 }
106 }
107
108 par(mfrow = c(3, 3))
109 plot.invgauss_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 10), func="dinvgauss")
110
111 par(mfrow = c(3, 3))
112 plot.invgauss_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pinvgauss")
113
114 par(mfrow = c(3, 3))
115 plot.invgauss_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sinvgauss")
116
117 par(mfrow = c(3, 3))
118 plot.invgauss_seq(x, m, s, xlim=c(min(x), max(x)), ylim=c(0, 25), func="hinvgauss")

```

14.27.5 Half-normal Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} \sqrt{\frac{2}{\pi}} e^{-\frac{(t-a)^2}{2b^2}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$2\Phi\left(\frac{a-t}{b}\right)$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{2\sqrt{2\pi}} \frac{e^{-\frac{(t-a)^2}{2b^2}}}{\Phi\left(\frac{a-t}{b}\right)}$	IHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 52: Half-normal 분포함수에 기반한 척도 함수

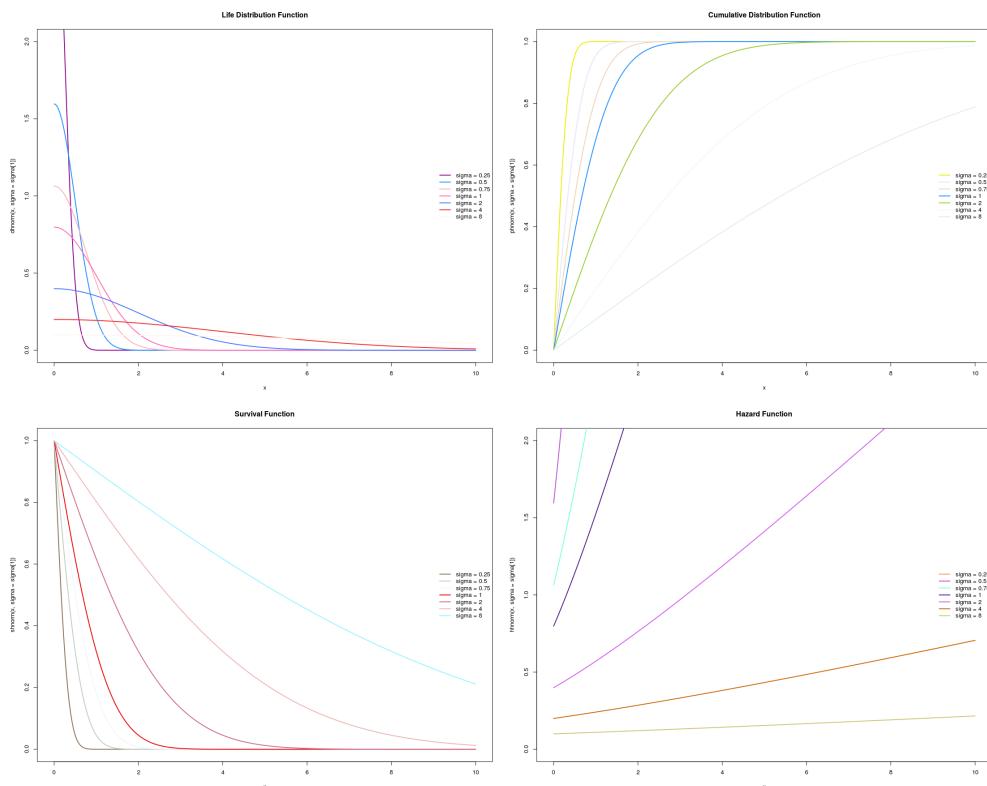


Figure 109: Half-normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 41. Half-normal Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(extraDistr)
4
5 ##### Chi-square Distribution
6 ### parameter
7 sigma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8
9 ### input varialbe
10 x = seq(0, 10, length.out = 1000)
11
12
13 ### 수명 분포
14 dhnorm(x, sigma = 1)
15
16
17 ### 분위수 함수
18 qhnorm(x, sigma = 1)
19
20
21 ### 난수 함수
22 rhnorm(x, sigma = 1)
23
24
25 ### 누적분포함수
26 phnorm(x, sigma = 1)
27
28
29 ### 생존함수
30 shnorm = function (x, sigma = 1)
31 {
32   fx = 1 - phnorm(x, sigma = sigma)
33   return(fx)
34 }
35
36
37
38 ### 위험함수
39 hhnorm = function (x, sigma = 1)
40 {
41   fx = dhnorm(x, sigma = sigma) / shnorm(x, sigma = sigma)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.hnorm_seq = function(x, sigma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dhnorm")
51 {
52   color=colorPalette(300)
53
54   len_sigma = length(sigma) # sigma 파라메터의 길이
55
56   color_counter = 1
57   color_counter_init = color_counter
58   legend_name = NULL;
59
60
61   if (func=="dhnorm") # 수명분포
62   {
63     plot(x, dhnorm(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function
64     ")
65     for (i in 1:len_sigma) ### 파라메터: sigma
66     {
67       lines(x, dhnorm(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
68       color_counter = color_counter + 1;
69       legend_name = c(legend_name, paste("sigma = ", sigma[i], sep=""))
70     }
71   else if (func == "phnorm") # 누적분포함수
72   {
73     plot(x, phnorm(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution
74     Function")
75     for (i in 1:len_sigma) ### 파라메터: sigma
76     {
77       lines(x, phnorm(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
78       color_counter = color_counter + 1;
79       legend_name = c(legend_name, paste("sigma = ", sigma[i], sep=""))
80     }
81   else if (func == "shnorm") # 생존함수
82   {
83     plot(x, shnorm(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
84     for (i in 1:len_sigma) ### 파라메터: sigma
```

```

85     {
86         lines(x, shnorm(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
87         color_counter = color_counter + 1;
88         legend_name = c(legend_name, paste("sigma = ", sigma[i], sep=""))
89     }
90 }
91 else if (func == "hhnorm") # 위험함수
92 {
93     plot(x, hhnorm(x, sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
94     for (i in 1:len_sigma) ### 파라미터: sigma
95     {
96         lines(x, hhnorm(x, sigma=sigma[i]), col=color[color_counter], lwd=2);
97         color_counter = color_counter + 1;
98         legend_name = c(legend_name, paste("sigma = ", sigma[i], sep=""))
99     }
100 }
101 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
102 }
103
104 par(mfrow = c(2, 2))
105 plot.hnorm_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dhnorm")
106 plot.hnorm_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 1), func="phnorm")
107 plot.hnorm_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 1), func="shnorm")
108 plot.hnorm_seq(x, sigma, xlim=c(min(x), max(x)), ylim=c(0, 2), func="hhnorm")

```

14.27.6 Trimmed(Truncated) Normal Distribution(절사 정규 분포)

Table 53: 절사정규분포 함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$\begin{cases} 0 & t < t_0 \\ \frac{1-\Phi(\frac{t-\mu}{\sigma})}{\sqrt{2\pi}\sigma[1-\Phi(\frac{t_0-\mu}{\sigma})]} & t \geq 0 \end{cases}$
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\frac{1-\Phi(\frac{t-\mu}{\sigma})}{1-\Phi(\frac{t_0-\mu}{\sigma})} \quad t \geq 0$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{e^{-\frac{1}{2}(\frac{t-\mu}{\sigma})}}{\sqrt{2\pi}\sigma[1-\Phi(\frac{t-\mu}{\sigma})]} \quad t \geq 0$
변수		$-\infty \leq t \leq \infty$
파라메터		$\mu, \sigma > 0$

Code 42. 절사정규분포에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 ##### 절사 정규 분포 (작성중)
2 require(truncnorm)
3 dtruncnorm(x, a=-Inf, b=Inf, mean = 0, sd = 1)
4 ptruncnorm(q, a=-Inf, b=Inf, mean = 0, sd = 1)
5 qtruncnorm(p, a=-Inf, b=Inf, mean = 0, sd = 1)
6 rtruncnorm(n, a=-Inf, b=Inf, mean = 0, sd = 1)
7etruncnorm(a=-Inf, b=Inf, mean=0, sd=1)
8 vtruncnorm(a=-Inf, b=Inf, mean=0, sd=1)

```

14.28 Parabolic U-shaped Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{3}{2b} \left(\frac{t-a}{b}\right)^2$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\frac{1}{2} \left[1 - \left(\frac{t-a}{b}\right)^3\right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{3(a-t)^2}{b^3 + (a-t)^3}$	DIHR with minimum at $t^* = a$ and $h(t^*) = 0$
누적 위험률 함수 (누적 고장률 함수)	$H(t) = -\log S(t)$	작성중	
변수		$a - b \leq t \leq a + b$	
파라미터		$a \in \mathbf{R}, b > 0$	

Table 54: Parabolic U-shaped 분포함수에 기반한 척도 함수

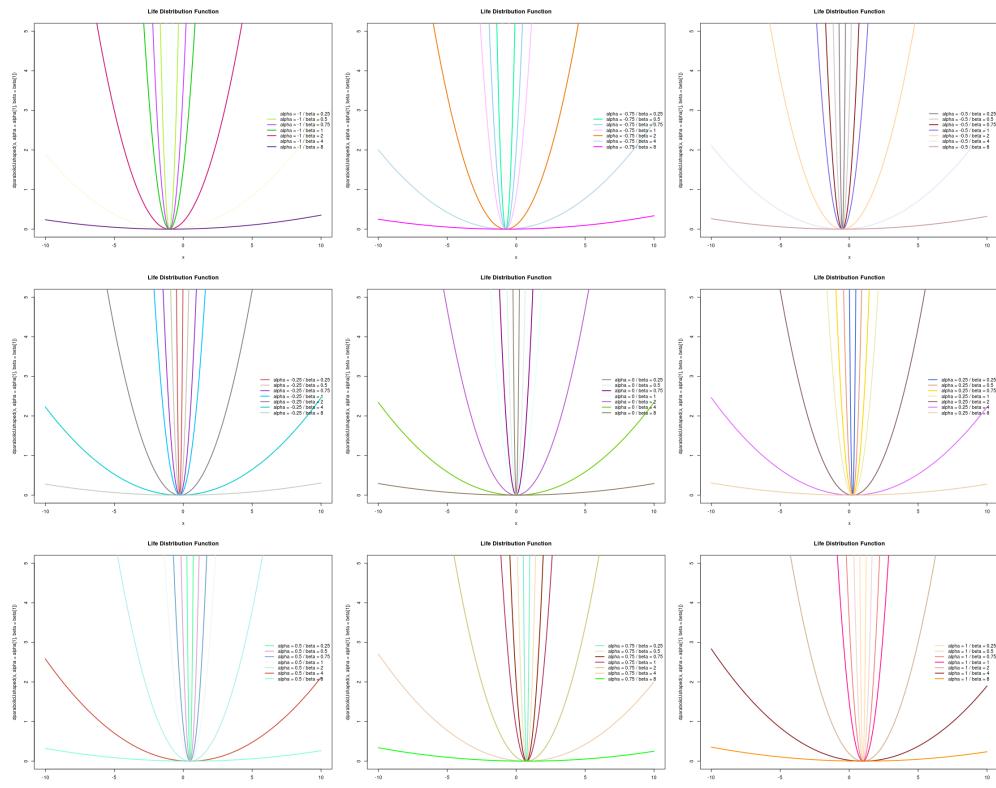


Figure 110: Parabolic U-shaped Distribution에 기반한 수명분포함수

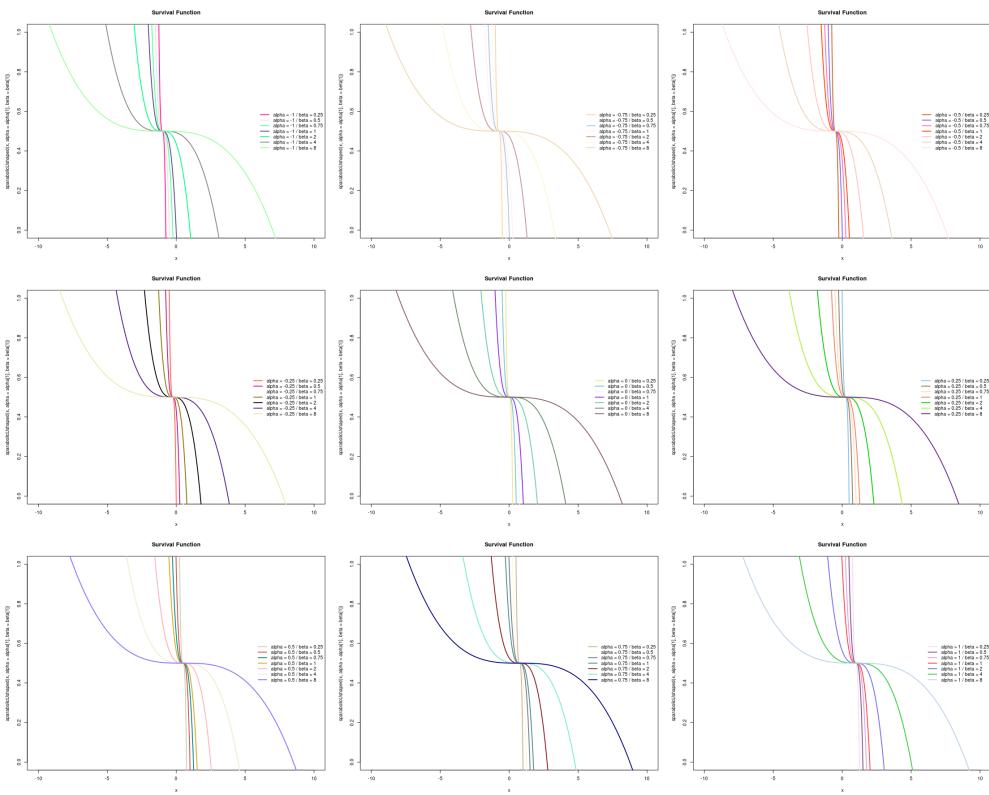


Figure 111: Parabolic U-shaped Distribution에 기반한 생존함수

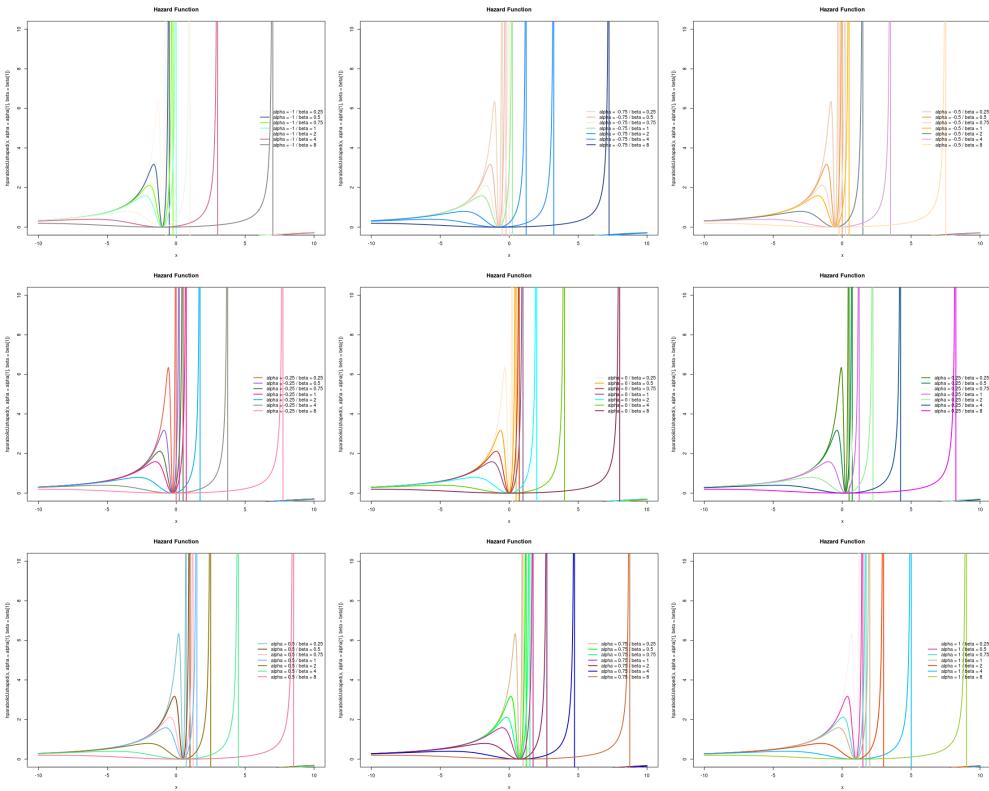


Figure 112: Parabolic U-shaped Distribution에 기반한 위험함수

Code 43. Parabolic U-shaped Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### parabolicUshaped Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dparabolicUshaped = function(x, alpha = 0, beta = 1)
16 {
17   fx = (3/(2*beta)) * ((x - alpha)/beta)^2
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 pparabolicUshaped = function(x, alpha = 0, beta = 1)
24 {
25   fx = -(dparabolicUshaped(x, alpha = alpha, beta = beta) - 1)
26   return(fx)
27 }
28
29
30 ### 생존함수
31 sparabolicUshaped = function (x, alpha = 0, beta = 1)
32 {
33   fx = (1/2) * (1 - ((x-alpha)/beta)^3)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hparabolicUshaped = function (x, alpha = 0, beta = 1)
40 {
41   fx = dparabolicUshaped(x, alpha, beta) / sparabolicUshaped(x, alpha, beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.parabolicUshaped_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dparabolicUshaped")
51 {
52   color=colorPalette(300)
53
54   len_alpha = length(alpha) # alpha 파라메터의 길이
55   len_beta = length(beta) # beta 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_alpha) ### 파라메터: alpha
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dparabolicUshaped") # 수명분포
64     {
65       plot(x, dparabolicUshaped(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
66         ="Life Distribution Function")
67       for (j in 1:len_beta) ### 파라메터: beta
68       {
69         lines(x, dparabolicUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
72       }
73     }
74     else if (func == "pparabolicUshaped") # 누적분포함수
75     {
76       plot(x, pparabolicUshaped(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
77         ="Cumulative Distribution Function")
78       for (j in 1:len_beta) ### 파라메터: beta
79       {
80         lines(x, pparabolicUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
83     }
84     else if (func == "sparabolicUshaped") # 생존함수
85     {

```

```

85     plot(x, sparabolicUshaped(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
86             ="Survival Function")
87     for (j in 1:len_beta) ### 파라미터: beta
88     {
89         lines(x, sparabolicUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
90         color_counter = color_counter + 1;
91         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
92     }
93   } else if (func == "hparabolicUshaped") # 위험함수
94   {
95     plot(x, hparabolicUshaped(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
96             ="Hazard Function")
97     for (j in 1:len_beta) ### 파라미터: beta
98     {
99         lines(x, hparabolicUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100        color_counter = color_counter + 1;
101        legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102    }
103  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.parabolicUshaped_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dparabolicUshaped")
109
110 par(mfrow = c(3, 3))
111 plot.parabolicUshaped_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pparabolicUshaped")
112
113 par(mfrow = c(3, 3))
114 plot.parabolicUshaped_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sparabolicUshaped")
115
116 par(mfrow = c(3, 3))
117 plot.parabolicUshaped_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hparabolicUshaped")

```

14.28.1 Parabolic Inverted U-shaped Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{3}{4b} \left[1 - \left(\frac{t-a}{b} \right)^2 \right]$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\frac{1}{2} - \frac{1}{4} \left[3 \left(\frac{t-a}{b} \right) - \left(\frac{t-a}{b} \right)^3 \right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	Complicated form	IHR
변수 파라메터		$a - b \leq t \leq a + b$	
		$a \in \mathbf{R}, b > 0$	

Table 55: Parabolic Inverted U-shaped 분포함수에 기반한 척도 함수

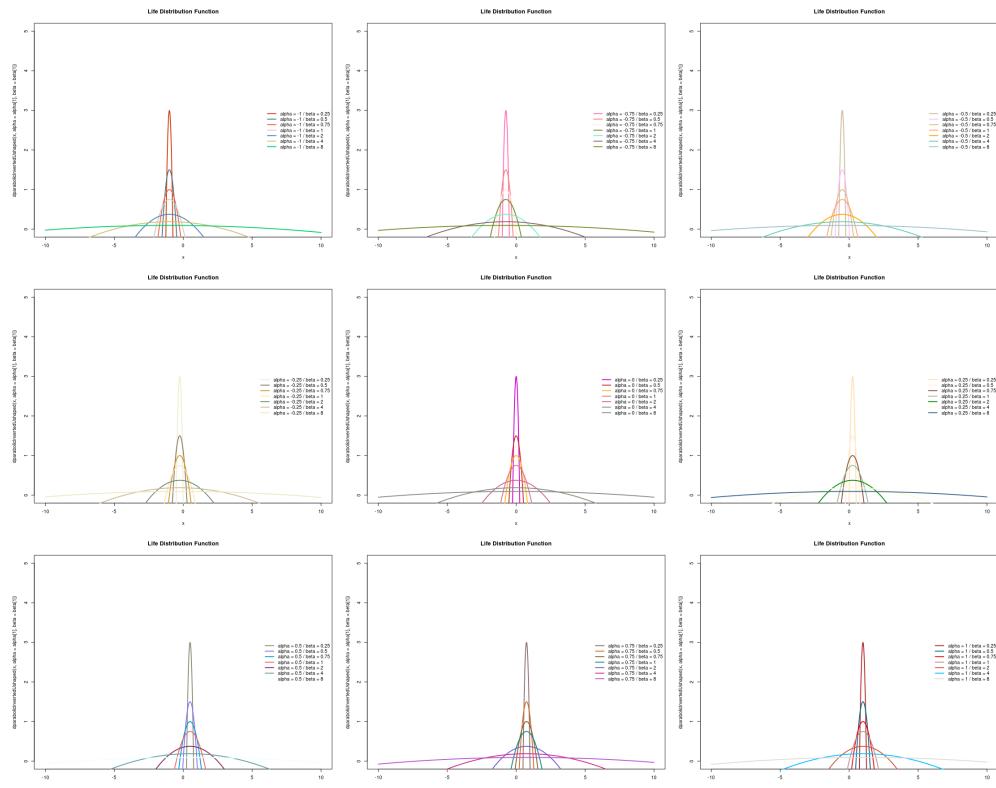


Figure 113: Parabolic Inverted U-shaped Distribution에 기반한 수명분포함수

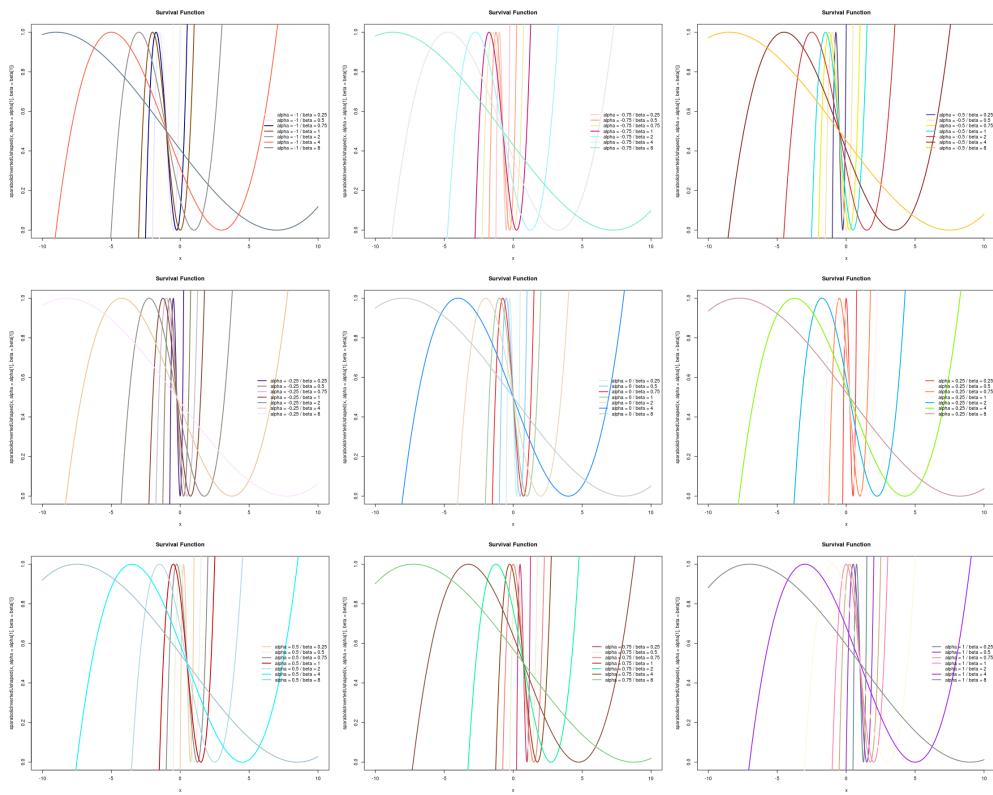


Figure 114: Parabolic Inverted U-shaped Distribution에 기반한 생존함수

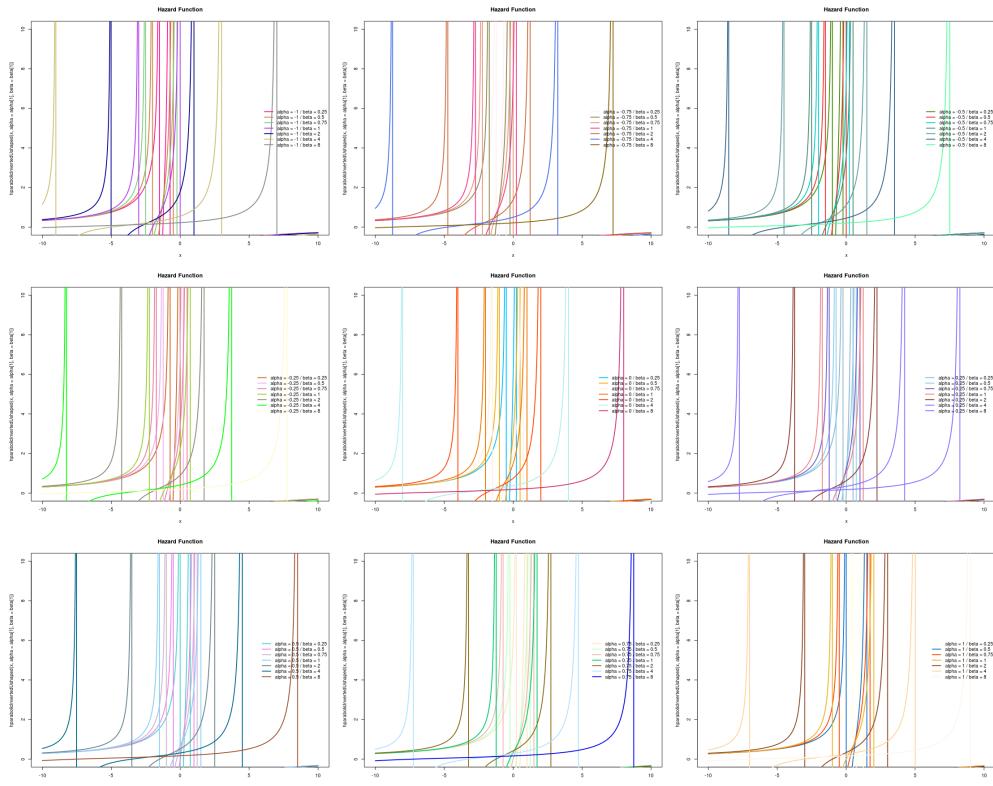


Figure 115: Parabolic Inverted U-shaped Distribution에 기반한 위험함수

Code 44. Parabolic Inverted U-shaped Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### parabolicInvertedUshaped Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dparabolicInvertedUshaped = function(x, alpha = 0, beta = 1)
16 {
17   fx = (3/(4*beta)) * (1 - ((x - alpha)/beta)^2)
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 pparabolicInvertedUshaped = function(x, alpha = 0, beta = 1)
24 {
25   fx = -(dparabolicInvertedUshaped(x, alpha = alpha, beta = beta) - 1)
26   return(fx)
27 }
28
29
30 ### 생존함수
31 sparabolicInvertedUshaped = function (x, alpha = 0, beta = 1)
32 {
33   fx = (1/2) - (1/4) * (3*((x-alpha)/beta) - ((x-alpha)/beta)^3)
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hparabolicInvertedUshaped = function (x, alpha = 0, beta = 1)
40 {
41   fx = dparabolicInvertedUshaped(x, alpha, beta) / sparabolicInvertedUshaped(x, alpha, beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.parabolicInvertedUshaped_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dparabolicInvertedUshaped"
51 )
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라메터의 길이
56   len_beta = length(beta) # beta 파라메터의 길이
57
58   color_counter = 1
59   for (i in 1:len_alpha) ### 파라메터: alpha
60   {
61     color_counter_init = color_counter
62     legend_name = NULL;
63
64     if (func=="dparabolicInvertedUshaped") # 수명분포
65     {
66       plot(x, dparabolicInvertedUshaped(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
67         "n", main="Life Distribution Function")
68       for (j in 1:len_beta) ### 파라메터: beta
69       {
70         lines(x, dparabolicInvertedUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
71         color_counter = color_counter + 1;
72         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
73       }
74     }
75     else if (func == "pparabolicInvertedUshaped") # 누적분포함수
76     {
77       plot(x, pparabolicInvertedUshaped(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
78         "n", main="Cumulative Distribution Function")
79       for (j in 1:len_beta) ### 파라메터: beta
80       {
81         lines(x, pparabolicInvertedUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
82         color_counter = color_counter + 1;
83         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
84       }
85     }
86     else if (func == "sparabolicInvertedUshaped") # 생존함수
87   }
88 }
```

```

84 {
85   plot(x, sparabolicInvertedUshaped(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = ,
86   n', main="Survival Function")
87   for (j in 1:len_beta) ### 파라미터: beta
88   {
89     lines(x, sparabolicInvertedUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
90     color_counter = color_counter + 1;
91     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
92   }
93 } else if (func == "hparabolicInvertedUshaped") # 위험함수
94 {
95   plot(x, hparabolicInvertedUshaped(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = ,
96   n', main="Hazard Function")
97   for (j in 1:len_beta) ### 파라미터: beta
98   {
99     lines(x, hparabolicInvertedUshaped(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102  }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106 par(mfrow = c(3, 3))
107 plot.parabolicInverted_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dparabolicInvertedUshaped")
108
109 par(mfrow = c(3, 3))
110 plot.parabolicInverted_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pparabolicInvertedUshaped")
111
112 par(mfrow = c(3, 3))
113 plot.parabolicInverted_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sparabolicInvertedUshaped")
114
115 par(mfrow = c(3, 3))
116 plot.parabolicInverted_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hparabolicInvertedUshaped")
117

```

14.29 Pareto Distribution of the first kind

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(\frac{t-a}{b}\right)^{-(c+1)}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$\left(\frac{t-a}{b}\right)^{-c}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c}{t-a}$	DHR
변수		$t \geq a + b$	
파라미터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 56: Pareto 분포함수에 기반한 척도 함수

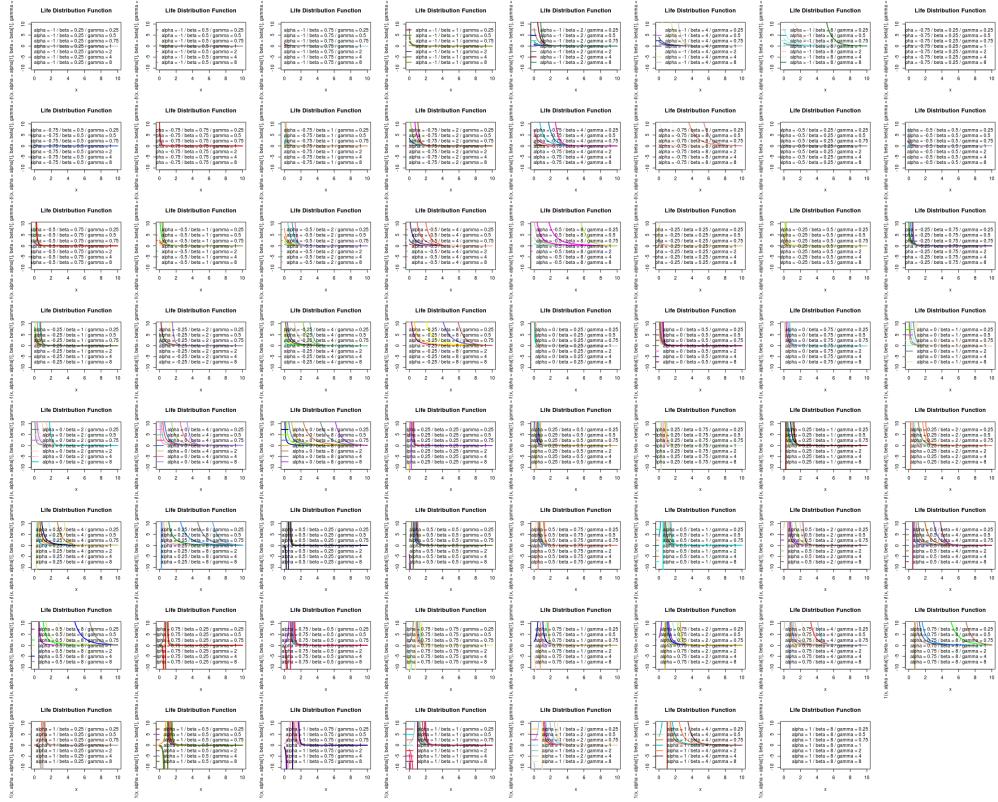


Figure 116: Pareto Distribution에 기반한 수명분포함수



Figure 117: Pareto Distribution에 기반한 생존함수



Figure 118: Pareto Distribution에 기반한 위험함수

Code 45. Pareto Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Pareto Distribution of the first kind
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dpareto1 = function(x, alpha = 1, beta = 1, gamma = 1)
17 {
18   fx = (gamma/beta) * (((x-alpha)/beta)^(-(gamma+1)))
19   return(fx)
20 }
21
22
23 ### 누적분포함수
24 ppareto1 = function(x, alpha = 1, beta = 1, gamma = 1)
25 {
26   fx = -(spareto1(x, alpha, beta, gamma) - 1)
27   return(fx)
28 }
29
30
31 ### 생존함수
32 spareto1 = function (x, alpha = 1, beta = 1, gamma = 1)
33 {
34   fx = ((x-alpha)/beta)^(-gamma)
35   return(fx)
36 }
37
38
39 ### 위험함수
40 hpareto1 = function (x, alpha = 1, beta = 1, gamma = 1)
41 {
42   fx = dpareto1(x, alpha, beta, gamma) / spareto1(x, alpha, beta, gamma)
43   return(fx)
44 }
45
46
47
48
49
50 ##### Plot
51 plot.pareto1_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dpareto1")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라미터의 길이
56   len_beta = length(beta) # beta 파라미터의 길이
57   len_gamma = length(gamma) # gamma 파라미터의 길이
58
59   color_counter = 1
60   for (i in 1:len_alpha) ### 파라미터: alpha
61   {
62     if (func=="dpareto1") # 수명분포
63     {
64       for (j in 1:len_beta) ### 파라미터: beta
65       {
66         color_counter_init = color_counter
67         legend_name = NULL;
68         plot(x, dpareto1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
69         = 'n', main="Life Distribution Function")
70         for (k in 1:len_gamma) ### 파라미터: gamma
71         {
72           lines(x, dpareto1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
73           color_counter = color_counter + 1;
74           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
75         }
76         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
77       }
78     else if (func == "ppareto1") # 누적분포함수
79     {
80       for (j in 1:len_beta) ### 파라미터: beta
81     {
82       color_counter_init = color_counter
83       legend_name = NULL;
84       plot(x, ppareto1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
85       = 'n', main="Cumulative Distribution Function")
86       for (k in 1:len_gamma) ### 파라미터: gamma
87     }
88   }
89 }
```

```

86      }
87      lines(x, ppareto1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
88      color_counter = color_counter + 1;
89      legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
90    }
91    legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
92  }
93  else if (func == "spareto1") # 생존함수
94  {
95    for (j in 1:len_beta) ### 파라미터: beta
96    {
97      color_counter_init = color_counter
98      legend_name = NULL;
99      plot(x, spareto1(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
100        = 'n', main="Survival Function")
101      for (k in 1:len_gamma) ### 파라미터: gamma
102      {
103        lines(x, spareto1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
104        color_counter = color_counter + 1;
105        legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
106      }
107      legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
108    }
109  }
110  else if (func == "hpareto1") # 위험함수
111  {
112    for (j in 1:len_beta) ### 파라미터: beta
113    {
114      color_counter_init = color_counter
115      legend_name = NULL;
116      plot(x, hpareto1(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type
117        = 'n', main="Hazard Function")
118      for (k in 1:len_gamma) ### 파라미터: gamma
119      {
120        lines(x, hpareto1(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
121        color_counter = color_counter + 1;
122        legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
123      }
124      legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
125    }
126  }
127}
128
129 par(mfrow = c(8, 8))
130 plot.pareto1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dpareto1")
131
132 par(mfrow = c(8, 8))
133 plot.pareto1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="ppareto1")
134
135 par(mfrow = c(8, 8))
136 plot.pareto1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="spareto1")
137
138 par(mfrow = c(8, 8))
139 plot.pareto1_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hpareto1")

```

14.29.1 Generalized Pareto Distribution of the first kind

- o] 분포는 Generalized Lomax 분포로도 알려져 있다. Chapter 14.23.1를 참고하도록 하자.

14.30 Power Function Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(\frac{t-a}{b}\right)^{c-1}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \left(\frac{t-a}{b}\right)^c$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c}{a-t} \left[1 + \left[\left(\frac{t-a}{b}\right)^c - 1\right]^{-1}\right]$	DIHR for $0 < c < 1$ IHR for $c \geq 1$
변수 파라미터		$a \leq t \leq a+b$	
		$a \in \mathbf{R}, b > 0, c > 0$	

Table 57: Power Function Distribution에 기반한 척도 함수

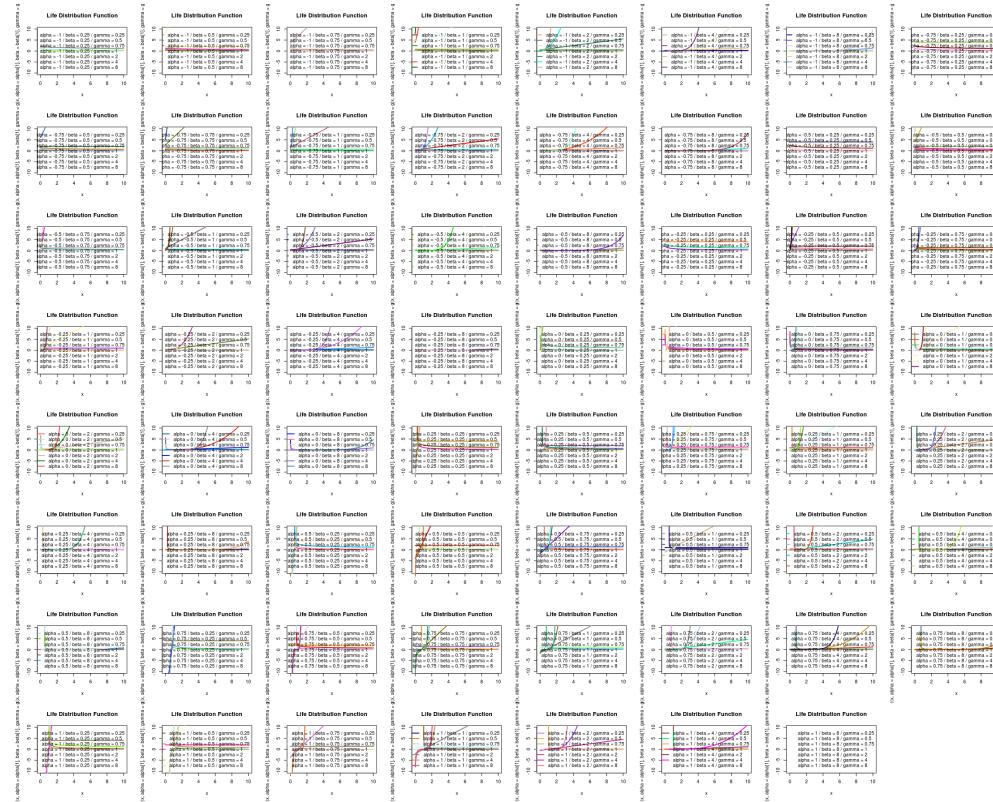


Figure 119: Power Function Distribution에 기반한 수명분포함수



Figure 120: Power Function Distribution에 기반한 생존함수

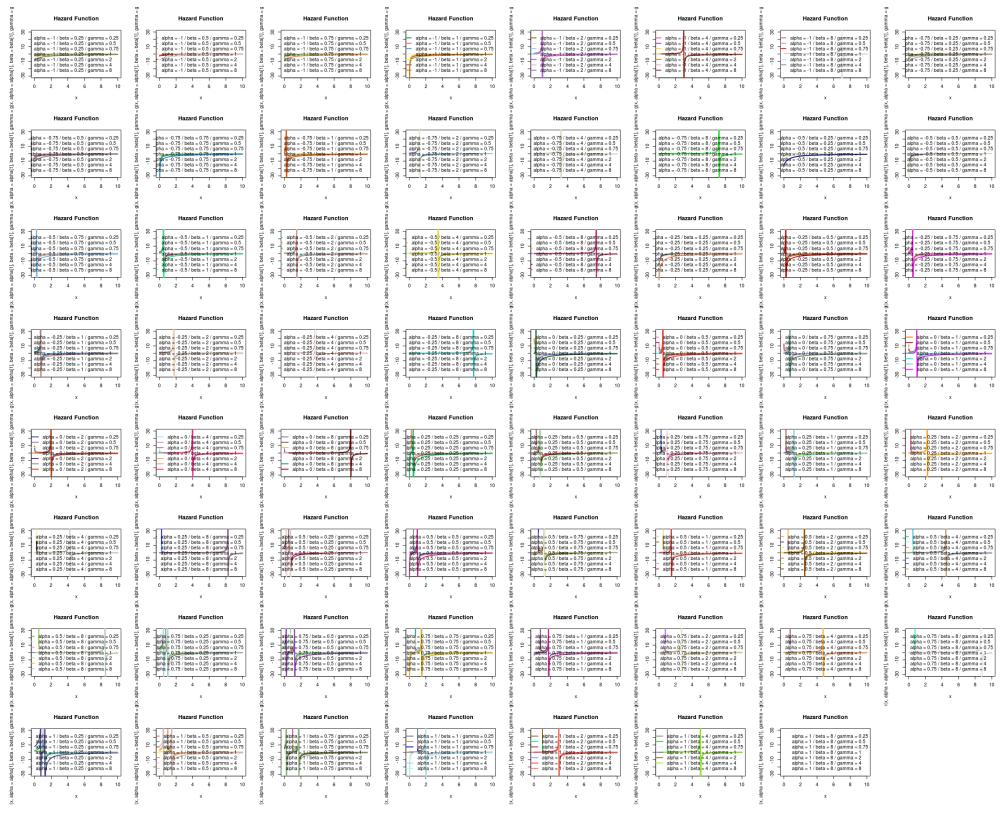


Figure 121: Power Function Distribution에 기반한 위험함수

Code 46. Power Function Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Pareto Distribution of the first kind
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dpower = function(x, alpha = 1, beta = 1, gamma = 1)
17 {
18   fx = (gamma/beta) * (((x - alpha)/beta)^(gamma-1))
19   return(fx)
20 }
21
22
23 ### 누적분포함수
24 ppower = function(x, alpha = 1, beta = 1, gamma = 1)
25 {
26   fx = -(spower(x, alpha, beta, gamma) - 1)
27   return(fx)
28 }
29
30
31 ### 생존함수
32 spower = function (x, alpha = 1, beta = 1, gamma = 1)
33 {
34   fx = 1 - (((x-alpha)/beta)^gamma)
35   return(fx)
36 }
37
38
39 ### 위험함수
40 hpower = function (x, alpha = 1, beta = 1, gamma = 1)
41 {
42   fx = dpower(x, alpha, beta, gamma) / spower(x, alpha, beta, gamma)
43   return(fx)
44 }
45
46
47
48
49
50 ##### Plot
51 plot.power_seq = function(x, alpha = 1, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dpower")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라미터의 길이
56   len_beta = length(beta) # beta 파라미터의 길이
57   len_gamma = length(gamma) # gamma 파라미터의 길이
58
59   color_counter = 1
60   for (i in 1:len_alpha) ### 파라미터: alpha
61   {
62     if (func=="dpower") # 수명분포
63     {
64       for (j in 1:len_beta) ### 파라미터: beta
65       {
66         color_counter_init = color_counter
67         legend_name = NULL;
68         plot(x, dpower(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
69           'n', main="Life Distribution Function")
70         for (k in 1:len_gamma) ### 파라미터: gamma
71         {
72           lines(x, dpower(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
73           color_counter = color_counter + 1;
74           legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
75         }
76         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
77       }
78     } else if (func == "ppower") # 누적분포함수
79     {
80       for (j in 1:len_beta) ### 파라미터: beta
81     {
82       color_counter_init = color_counter
83       legend_name = NULL;
84       plot(x, ppower(x, alpha=alpha[i], beta=beta[j], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
85           'n', main="Cumulative Distribution Function")
86       for (k in 1:len_gamma) ### 파라미터: gamma
87     }
88   }
89 }
```

```

86  }
87  {
88     lines(x, ppower(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
89     color_counter = color_counter + 1;
90     legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
91   )
92 }
93 }
94 else if (func == "spower") # 생존함수
95 {
96   for (j in 1:len_beta) ### 파라미터: beta
97   {
98     color_counter_init = color_counter
99     legend_name = NULL;
100    plot(x, spower(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
101      'n', main="Survival Function")
102    for (k in 1:len_gamma) ### 파라미터: gamma
103    {
104      lines(x, spower(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
105      color_counter = color_counter + 1;
106      legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
107    )
108  }
109 }
110 else if (func == "hpower") # 위험함수
111 {
112   for (j in 1:len_beta) ### 파라미터: beta
113   {
114     color_counter_init = color_counter
115     legend_name = NULL;
116     plot(x, hpower(x, alpha=alpha[1], beta=beta[1], gamma=gamma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type =
117       'n', main="Hazard Function")
118     for (k in 1:len_gamma) ### 파라미터: gamma
119     {
120       lines(x, hpower(x, alpha=alpha[i], beta=beta[j], gamma=gamma[k]), col=color[color_counter], lwd=2);
121       color_counter = color_counter + 1;
122       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], " / gamma = ", gamma[k], sep=""))
123     )
124   }
125 }
126 }
127 }
128 par(mfrow = c(8, 8))
129 plot.power_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dpower")
130
131 par(mfrow = c(8, 8))
132 plot.power_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="ppower")
133
134 par(mfrow = c(8, 8))
135 plot.power_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="spower")
136
137 par(mfrow = c(8, 8))
138 plot.power_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hpower")
139

```

14.31 Rayleigh Distribution(레이일리 분포)

Table 58: 레일리 분포함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$kte^{-\left(\frac{kt^2}{2}\right)}$
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{-\left(\frac{kt^2}{2}\right)}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	kt
변수		$t \geq 0$
파라메터		$k > 0$

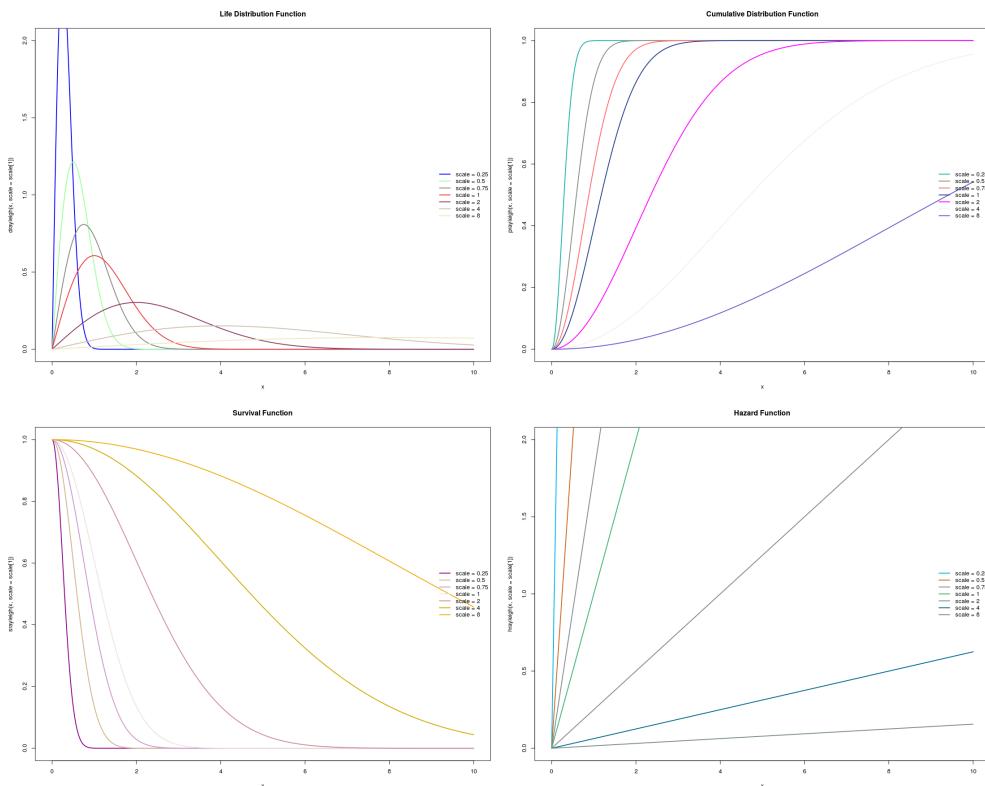


Figure 122: Rayleigh Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 47. Rayleigh Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Rayleigh Distribution
6 ### parameter
7 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8
9 ### input varialbe
10 x = seq(0, 10, length.out = 1000)
11
12
13 ### 수명 분포
14 drayleigh(x, scale = scale)
15
16
17 ### 분위수 함수
18 qrayleigh(x, scale = scale)
19
20
21 ### 난수 함수
22 rrayleigh(x, scale = scale)
23
24
25 ### 누적분포함수
26 prayleigh(x, scale = scale)
27
28
29 ### 생존함수
30 srayleigh = function (x, scale = 1)
31 {
32   fx = 1 - prayleigh(x, scale = scale)
33   return(fx)
34 }
35
36
37 ### 위험함수
38 hrayleigh = function (x, scale = 1)
39 {
40   fx = drayleigh(x, scale = scale) / srayleigh(x, scale = scale)
41   return(fx)
42 }
43
44
45
46
47
48 ##### Plot
49 plot.rayleigh_seq = function(x, scale = 1, xlim=c(0, 10), ylim=c(0, 5), func="drayleigh")
50 {
51   color=colorPalette(300)
52
53   len_scale = length(scale) # scale 파라메터의 길이
54
55   color_counter = 1
56   color_counter_init = color_counter
57   legend_name = NULL;
58
59
60   if (func=="drayleigh") # 수명분포
61   {
62     plot(x, drayleigh(x, scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution
63     Function")
64     for (i in 1:len_scale) ### 파라메터: scale
65     {
66       lines(x, drayleigh(x, scale=scale[i]), col=color[color_counter], lwd=2);
67       color_counter = color_counter + 1;
68       legend_name = c(legend_name, paste("scale = ", scale[i], sep=""))
69     }
70   }
71   else if (func == "prayleigh") # 누적분포함수
72   {
73     plot(x, prayleigh(x, scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution
74     Function")
75     for (i in 1:len_scale) ### 파라메터: scale
76     {
77       lines(x, prayleigh(x, scale=scale[i]), col=color[color_counter], lwd=2);
78       color_counter = color_counter + 1;
79       legend_name = c(legend_name, paste("scale = ", scale[i], sep=""))
80     }
81   }
82   else if (func == "srayleigh") # 생존함수
83   {
84     plot(x, srayleigh(x, scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
85     for (i in 1:len_scale) ### 파라메터: scale
86     {
```

```

85     lines(x, srayleigh(x, scale=scale[i]), col=color[color_counter], lwd=2);
86     color_counter = color_counter + 1;
87     legend_name = c(legend_name, paste("scale = ", scale[i], sep=""))
88   }
89 }
90 else if (func == "hrayleigh") # 위험함수
91 {
92   plot(x, hrayleigh(x, scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
93   for (i in 1:len_scale) ### 파라미터: scale
94   {
95     lines(x, hrayleigh(x, scale=scale[i]), col=color[color_counter], lwd=2);
96     color_counter = color_counter + 1;
97     legend_name = c(legend_name, paste("scale = ", scale[i], sep=""))
98   }
99 }
100 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
101 }
102
103 par(mfrow = c(2, 2))
104 plot.rayleigh_seq(x, scale, xlim=c(min(x), max(x)), ylim=c(0, 2), func="drayleigh")
105 plot.rayleigh_seq(x, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="prayleigh")
106 plot.rayleigh_seq(x, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="srayleigh")
107 plot.rayleigh_seq(x, scale, xlim=c(min(x), max(x)), ylim=c(0, 2), func="hrayleigh")

```

레일리 분포는 와이블 분포의 특수한 경우로, 와이블 분포의 형상 모수(shape parameter)가 2일 때 레일리 분포가 된다. 레일리분포는 마모 특성을 모형화하는 데 유용한 분포이다.

이 분포의 특징은, 위험률 함수(고장률 함수)가 원점을 지나며, 시간이 지남에 따라 선형적으로 증가한다는 것이다. 따라서, 레일리 분포의 위험률 함수(고장률 함수)는 $r(t) = kt^{\alpha}$ 이다.

14.31.1 Inverse Rayleigh Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{2b}{(t-a)^3} e^{-\frac{b}{(t-a)^3}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - e^{-\frac{b}{(t-a)^3}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$2b \left[(t-a)^3 \left(e^{\frac{b}{(t-a)^3}} - 1 \right) \right]^{-1}$	IDHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 59: Inverse Rayleigh 분포함수에 기반한 척도 함수

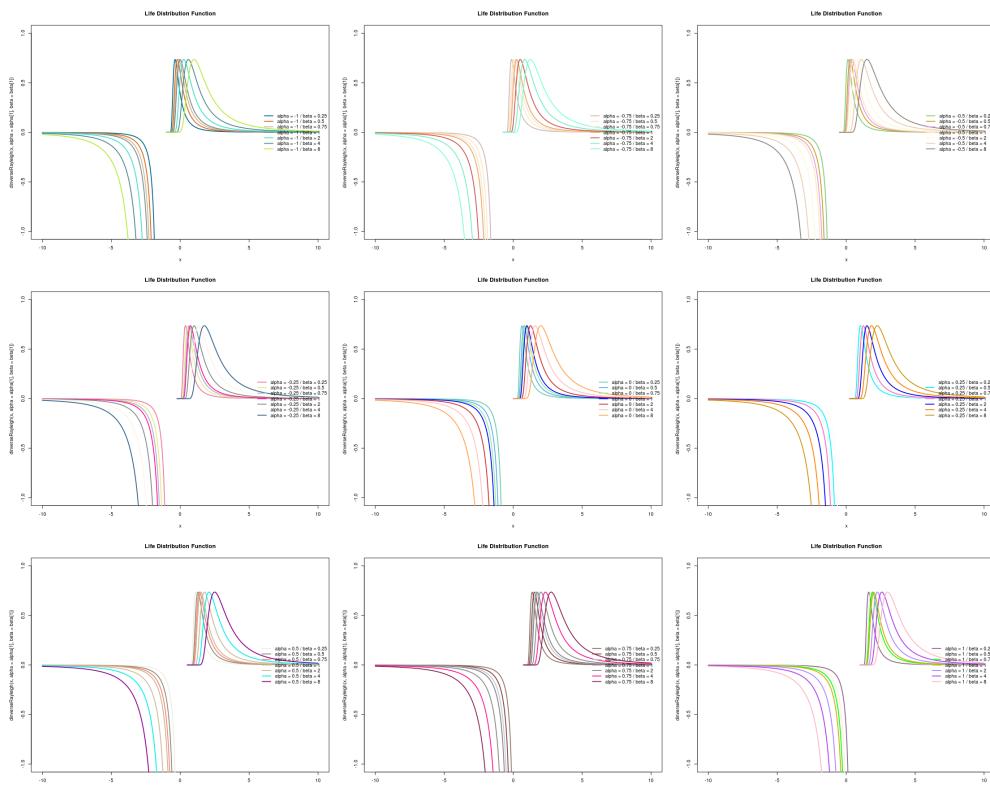


Figure 123: Inverse Rayleigh Distribution에 기반한 수명분포함수

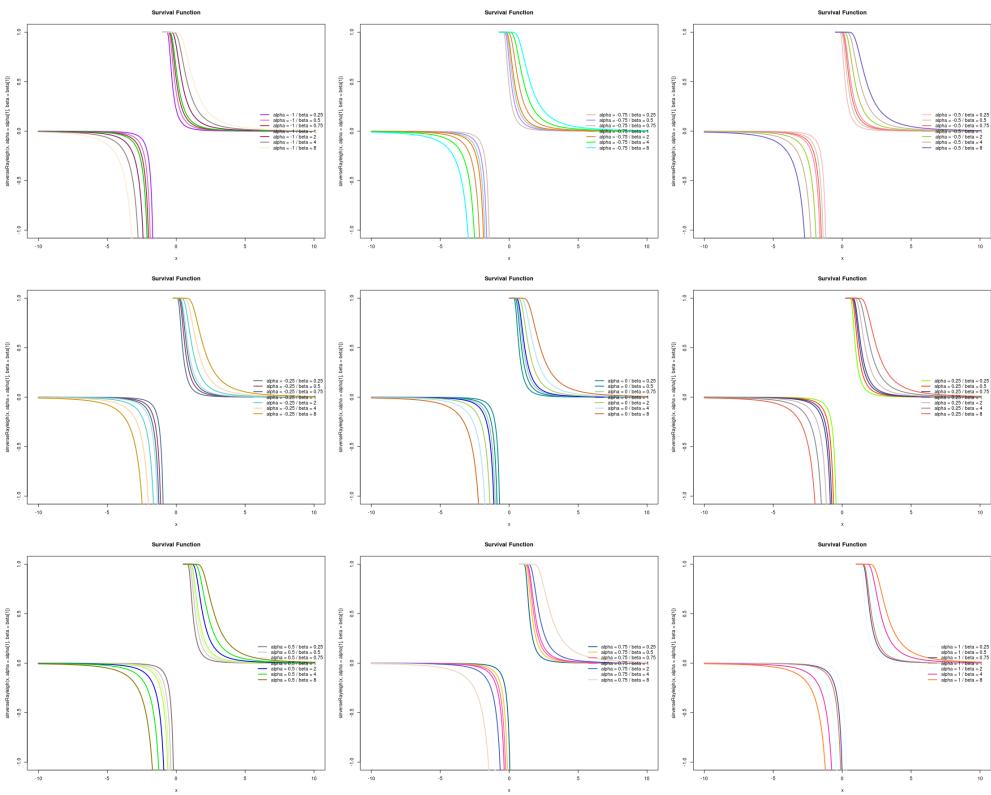


Figure 124: Inverse Rayleigh Distribution에 기반한 생존함수

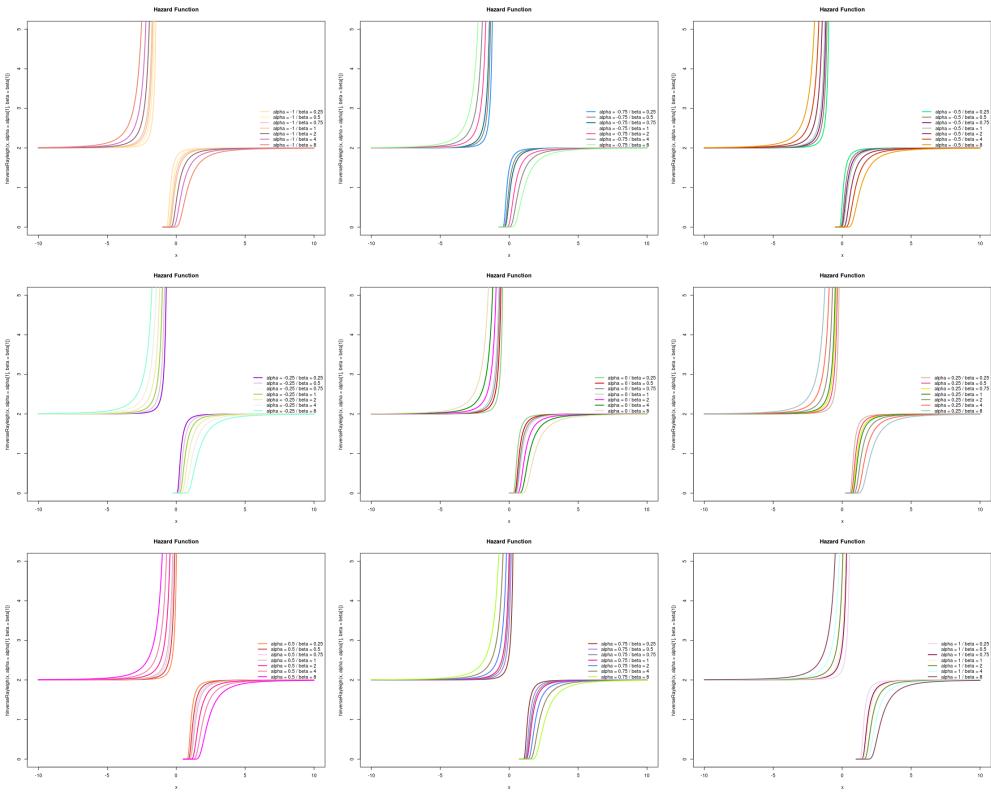


Figure 125: Inverse Rayleigh Distribution에 기반한 위험함수

Code 48. Inverse Rayleigh Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### inverseRayleigh Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dinverseRayleigh = function(x, alpha = 0, beta = 1)
16 {
17   fx = ((2 * beta)/((x-alpha)^3)) * exp(-(beta / ((x-alpha)^3)))
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 pinverseRayleigh = function(x, alpha = 0, beta = 1)
24 {
25   fx = -(sinverseRayleigh(x, alpha = alpha, beta = beta) - 1)
26   return(fx)
27 }
28
29
30 ### 생존함수
31 sinverseRayleigh = function (x, alpha = 0, beta = 1)
32 {
33   fx = 1 - exp(-(beta / ((x-alpha)^3)))
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hinverseRayleigh = function (x, alpha = 0, beta = 1)
40 {
41   fx = dinverseRayleigh(x, alpha, beta) / sinverseRayleigh(x, alpha, beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.inverseRayleigh_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dinverseRayleigh")
51 {
52   color=colorPalette(300)
53
54   len_alpha = length(alpha) # alpha 파라메터의 길이
55   len_beta = length(beta) # beta 파라메터의 길이
56
57   color_counter = 1
58   for (i in 1:len_alpha) ### 파라메터: alpha
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dinverseRayleigh") # 수명분포
64     {
65       plot(x, dinverseRayleigh(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
66         "Life Distribution Function")
67       for (j in 1:len_beta) ### 파라메터: beta
68       {
69         lines(x, dinverseRayleigh(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
72       }
73     }
74     else if (func == "pinverseRayleigh") # 누적분포함수
75     {
76       plot(x, pinverseRayleigh(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
77         "Cumulative Distribution Function")
78       for (j in 1:len_beta) ### 파라메터: beta
79       {
80         lines(x, pinverseRayleigh(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
83     }
84     else if (func == "sinverseRayleigh") # 생존함수
85     {
86       plot(x, sinverseRayleigh(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
87         "Survival Function")
88     }
89   }
90 }
```

```

86     for (j in 1:len_beta) ### 파라메터: beta
87     {
88       lines(x, sinverseRayleigh(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
91     }
92   }
93 else if (func == "hinverseRayleigh") # 위험함수
94 {
95   plot(x, hinverseRayleigh(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
96       "Hazard Function")
97   for (j in 1:len_beta) ### 파라메터: beta
98   {
99     lines(x, hinverseRayleigh(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102  }
103 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.inverseRayleigh_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 1), func="dinverseRayleigh")
109
110 par(mfrow = c(3, 3))
111 plot.inverseRayleigh_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 2), func="pinverseRayleigh")
112
113 par(mfrow = c(3, 3))
114 plot.inverseRayleigh_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 1), func="sinverseRayleigh")
115
116 par(mfrow = c(3, 3))
117 plot.inverseRayleigh_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="hinverseRayleigh")

```

14.31.2 Generalized Rayleigh Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$c \frac{t-a}{b^2} e^{-\frac{1}{2}(\frac{t-a}{b})^2} \left[1 - e^{-\frac{1}{2}(\frac{t-a}{b})^2}\right]^{c-1}$	
생존함수 (신뢰도함수)	$\begin{aligned} S(t) &= P(T > t) \\ &= 1 - P(T \leq t) \\ &= 1 - F(t) \\ &= e^{-H(t)} \end{aligned}$	$1 - \left[e^{-\frac{1}{2}(\frac{t-a}{b})^2}\right]^c$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	작성중	DIHR for $0 < c < 0.5$ IHR for $c \geq 0.5$
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 60: Generalized Rayleigh 분포함수에 기반한 척도 함수

$c = 1$ 이면 이 분포는 Rayleigh 분포가 된다.

14.32 Semi-elliptical Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{2}{b\pi} \sqrt{1 - \left(\frac{t-a}{b}\right)^2}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$\frac{1}{2} - \frac{1}{\pi} \left[\frac{t-a}{b} \sqrt{1 - \left(\frac{t-a}{b}\right)^2} + \arcsin\left(\frac{t-a}{b}\right) \right]$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$b \frac{4\sqrt{1 - \left(\frac{t-a}{b}\right)^2}}{\pi - 2 \left(\left(\frac{t-a}{b}\right) \sqrt{1 - \left(\frac{t-a}{b}\right)^2} + \arcsin\left(\frac{t-a}{b}\right) \right)}$	IHR
변수 파라미터		$a - b \leq t \leq a + b$ $a \in \mathbf{R}, b > 0$	

Table 61: Semi-elliptical 분포함수에 기반한 척도 함수

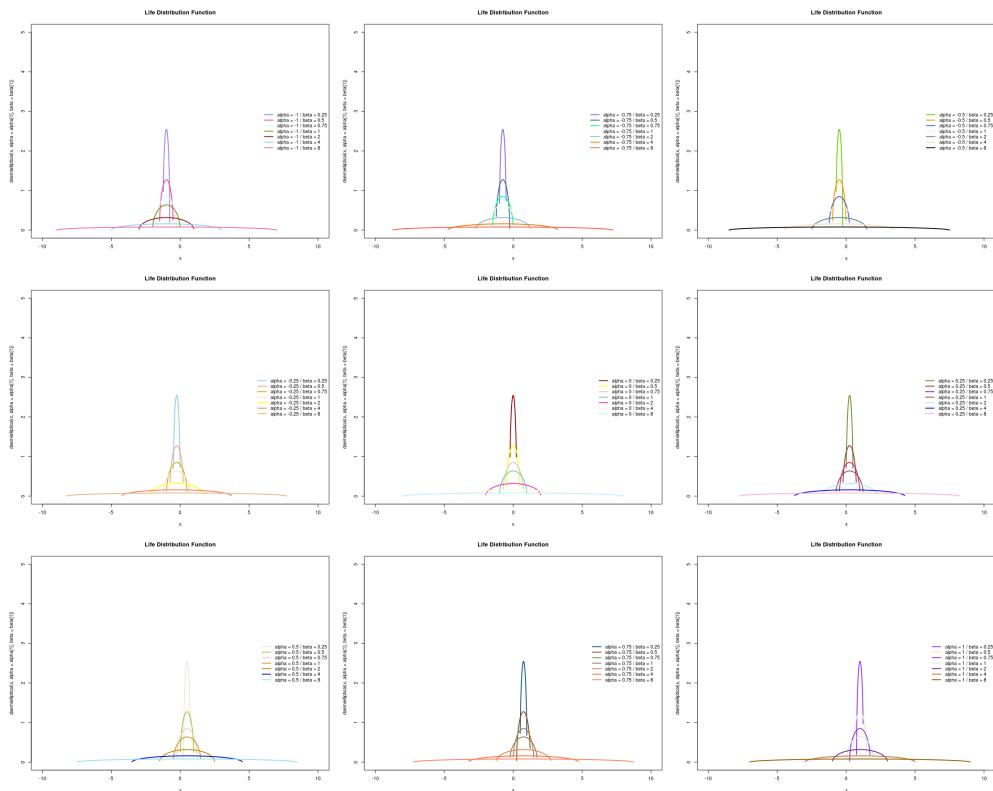


Figure 126: Semi-elliptical Distribution에 기반한 수명분포함수

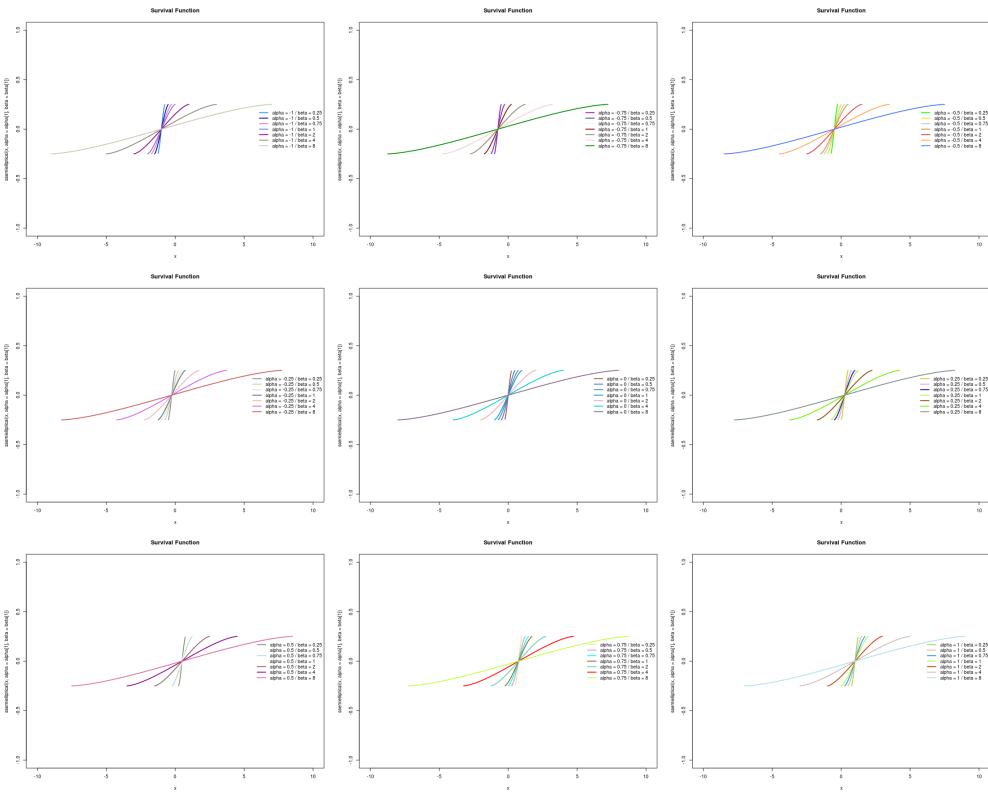


Figure 127: Semi-elliptical Distribution에 기반한 생존함수

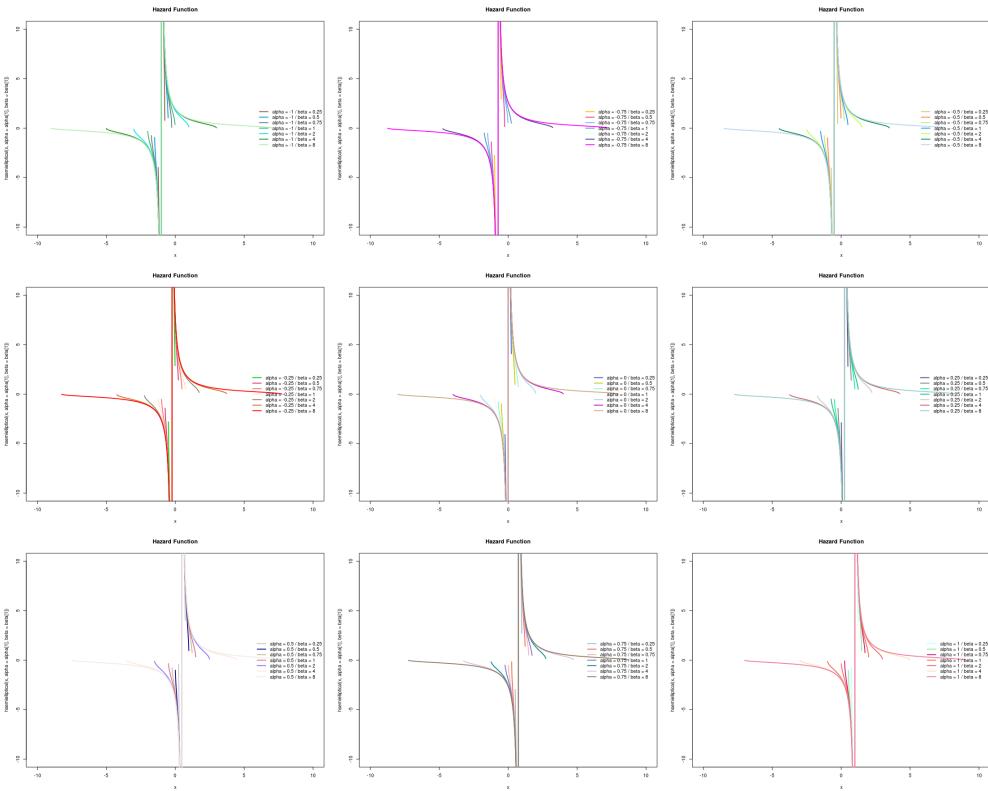


Figure 128: Semi-elliptical Distribution에 기반한 위험함수

Code 49. Semi-elliptical Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(pracma)
4
5 ###### semielliptical Distribution
6 #### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 #### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 #### 수명 분포
15 dsemielliptical = function(x, alpha = 0, beta = 1)
16 {
17   fx = (2/(beta * pi)) * sqrt(1 - ((x-alpha)/beta)^2)
18   return(fx)
19 }
20
21
22 #### 누적분포함수
23 psemielliptical = function(x, alpha = 0, beta = 1)
24 {
25   fx = -(ssemielliptical(x, alpha = alpha, beta = beta) - 1)
26   return(fx)
27 }
28
29
30 #### 생존함수
31 ssemielliptical = function (x, alpha = 0, beta = 1)
32 {
33   temp = (x - alpha)/beta
34   fx = (1/2) * (1/pi) * (temp * sqrt(1-temp^2) + asin(temp))
35   return(fx)
36 }
37
38
39 #### 위험함수
40 hsemielliptical = function (x, alpha = 0, beta = 1)
41 {
42   fx = dsemielliptical(x, alpha, beta) / ssemielliptical(x, alpha, beta)
43   return(fx)
44 }
45
46
47
48
49
50 ###### Plot
51 plot.semielliptical_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dsemielliptical")
52 {
53   color=colorPalette(300)
54
55   len_alpha = length(alpha) # alpha 파라메터의 길이
56   len_beta = length(beta) # beta 파라메터의 길이
57
58   color_counter = 1
59   for (i in 1:len_alpha) ### 파라메터: alpha
60   {
61     color_counter_init = color_counter
62     legend_name = NULL;
63
64     if (func=="dsemielliptical") # 수명분포
65     {
66       plot(x, dsemielliptical(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
67       for (j in 1:len_beta) ### 파라메터: beta
68       {
69         lines(x, dsemielliptical(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
72       }
73     }
74   else if (func == "psemielliptical") # 누적분포함수
75   {
76     plot(x, psemielliptical(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
77     for (j in 1:len_beta) ### 파라메터: beta
78     {
79       lines(x, psemielliptical(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
80       color_counter = color_counter + 1;
81       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
82     }
83   }
84   else if (func == "ssemielliptical") # 생존함수
85   {
86

```

```

87     plot(x, ssemielliptical(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
88     for (j in 1:len_beta) ### 파라미터: beta
89     {
90       lines(x, ssemielliptical(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
91       color_counter = color_counter + 1;
92       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
93     }
94   }
95   else if (func == "hsemielliptical") # 위험함수
96   {
97     plot(x, hsemielliptical(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
98     for (j in 1:len_beta) ### 파라미터: beta
99     {
100       lines(x, hsemielliptical(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
101       color_counter = color_counter + 1;
102       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
103     }
104   }
105   legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
106 }
107 }
108 par(mfrow = c(3, 3))
109 plot.semielliptical_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dsemielliptical")
110 par(mfrow = c(3, 3))
111 plot.semielliptical_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 2), func="psemielliptical")
112 par(mfrow = c(3, 3))
113 plot.semielliptical_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 1), func="ssemielliptical")
114 par(mfrow = c(3, 3))
115 plot.semielliptical_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="hsemielliptical")
116
117
118
119

```

14.33 t Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{\Gamma(\frac{\nu+1}{\nu})}{\sqrt{\pi\nu}\Gamma(\frac{\nu}{2})} \left(1 + \frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	No closed form	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	No closed form	
변수		$t \in \mathbf{R}$	
파라메터		$\nu > 0$	

Table 62: t 분포함수에 기반한 척도 함수

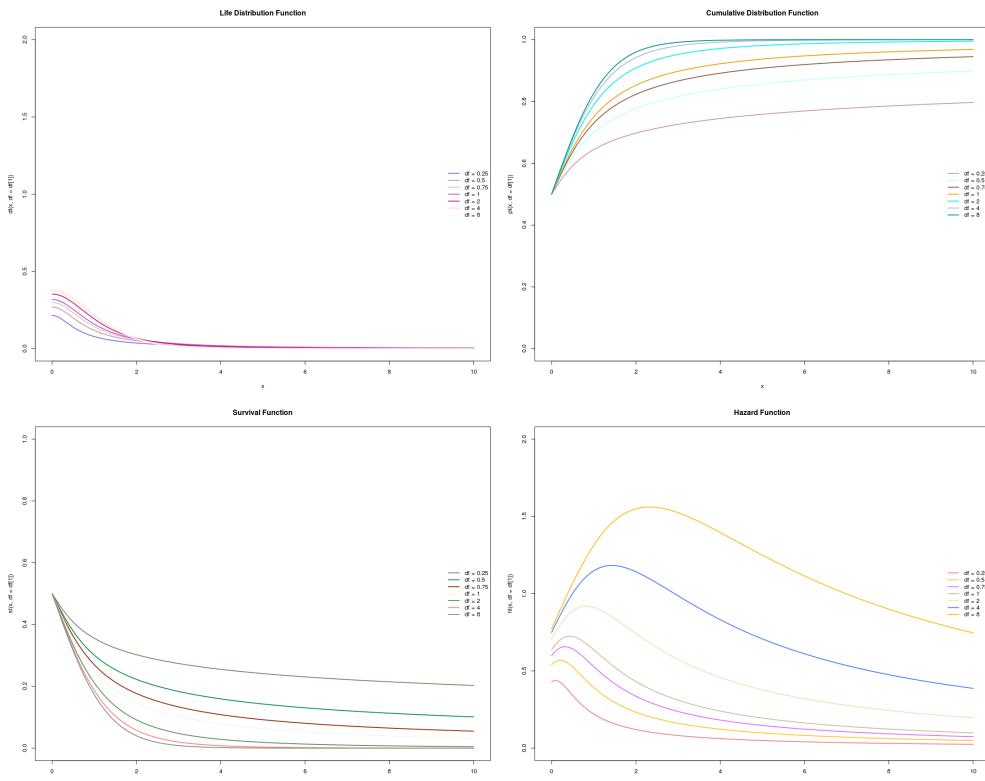


Figure 129: t Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률

Code 50. t Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```
1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### Chi-square Distribution
6 ### parameter
7 df = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8
9 ### input varialbe
10 x = seq(0, 10, length.out = 1000)
11
12
13 ### 수명 분포
14 dt(x, df = df)
15
16
17 ### 분위수 함수
18 qt(x, df = df)
19
20
21 ### 난수 함수
22 rt(x, df = df)
23
24
25 ### 누적분포함수
26 pt(x, df = df)
27
28
29 ### 생존함수
30 st = function (x, df = 1)
31 {
32   fx = 1 - pt(x, df = df)
33   return(fx)
34 }
35
36
37 ### 위험함수
38 ht = function (x, df = 1)
39 {
40   fx = dt(x, df = df) / st(x, df = df)
41   return(fx)
42 }
43
44
45
46
47
48 ##### Plot
49 plot.t_seq = function(x, df = 1, xlim=c(0, 10), ylim=c(0, 5), func="dt")
50 {
51   color=colorPalette(300)
52
53   len_df = length(df) # df 파라미터의 길이
54
55   color_counter = 1
56   color_counter_init = color_counter
57   legend_name = NULL;
58
59
60   if (func=="dt") # 수명분포
61   {
62     plot(x, dt(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life Distribution Function")
63     for (i in 1:len_df) ### 파라미터: df
64     {
65       lines(x, dt(x, df=df[i]), col=color[color_counter], lwd=2);
66       color_counter = color_counter + 1;
67       legend_name = c(legend_name, paste("df = ", df[i], sep=""))
68     }
69   }
70   else if (func == "pt") # 누적분포함수
71   {
72     plot(x, pt(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
73     for (i in 1:len_df) ### 파라미터: df
74     {
75       lines(x, pt(x, df=df[i]), col=color[color_counter], lwd=2);
76       color_counter = color_counter + 1;
77       legend_name = c(legend_name, paste("df = ", df[i], sep=""))
78     }
79   }
80   else if (func == "st") # 생존함수
81   {
82     plot(x, st(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
83     for (i in 1:len_df) ### 파라미터: df
84     {
85       lines(x, st(x, df=df[i]), col=color[color_counter], lwd=2);
86       color_counter = color_counter + 1;
```

```

87     legend_name = c(legend_name, paste("df = ", df[i], sep="")))
88   }
89 }
90 else if (func == "ht") # 위험함수
91 {
92   plot(x, ht(x, df=df[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
93   for (i in 1:len_df) ### 파라메터: df
94   {
95     lines(x, ht(x, df=df[i]), col=color[color_counter], lwd=2);
96     color_counter = color_counter + 1;
97     legend_name = c(legend_name, paste("df = ", df[i], sep="")))
98   }
99 }
100 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
101 }
102
103 par(mfrow = c(2, 2))
104 plot.t_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 2), func="dt")
105 plot.t_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pt")
106 plot.t_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 1), func="st")
107 plot.t_seq(x, df, xlim=c(min(x), max(x)), ylim=c(0, 2), func="ht")

```

14.34 Teisser Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} \left[e^{\frac{t-a}{b}} - 1 \right] e^{\left[1 + \frac{t-a}{b} - e^{\frac{t-a}{b}} \right]}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{1 + \frac{t-a}{b} - e^{\frac{t-a}{b}}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{b} \left[e^{\frac{t-a}{b}} - 1 \right]$	IHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 63: Teisser 분포함수에 기반한 척도 함수

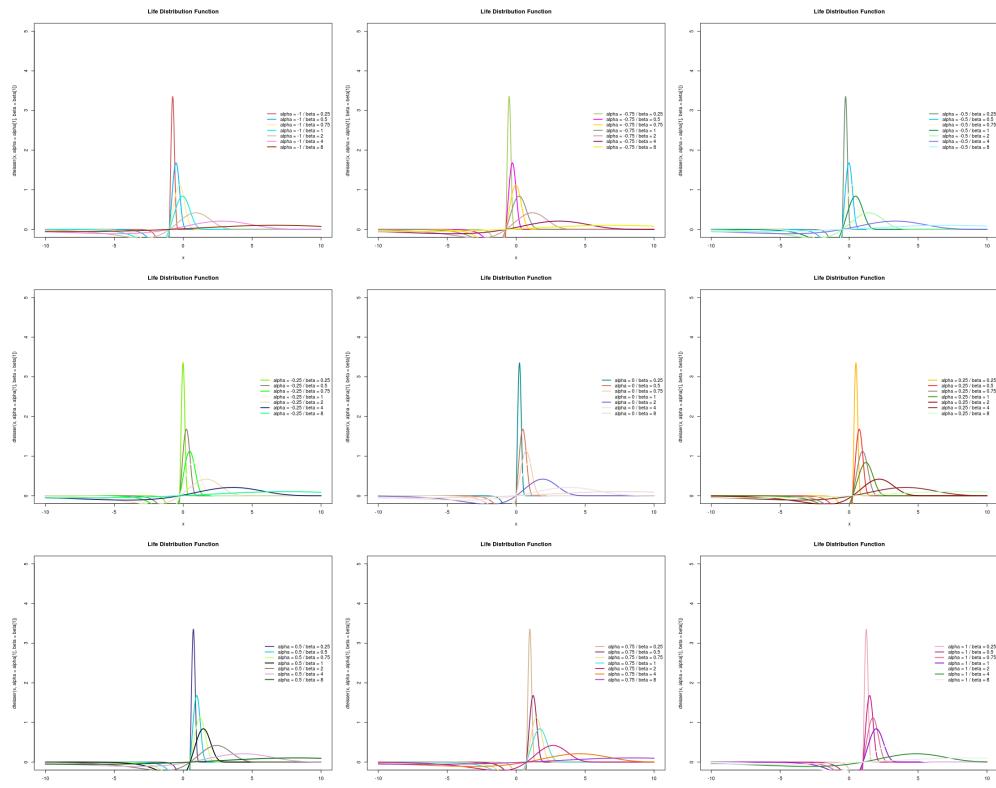


Figure 130: Teisser Distribution에 기반한 수명분포함수

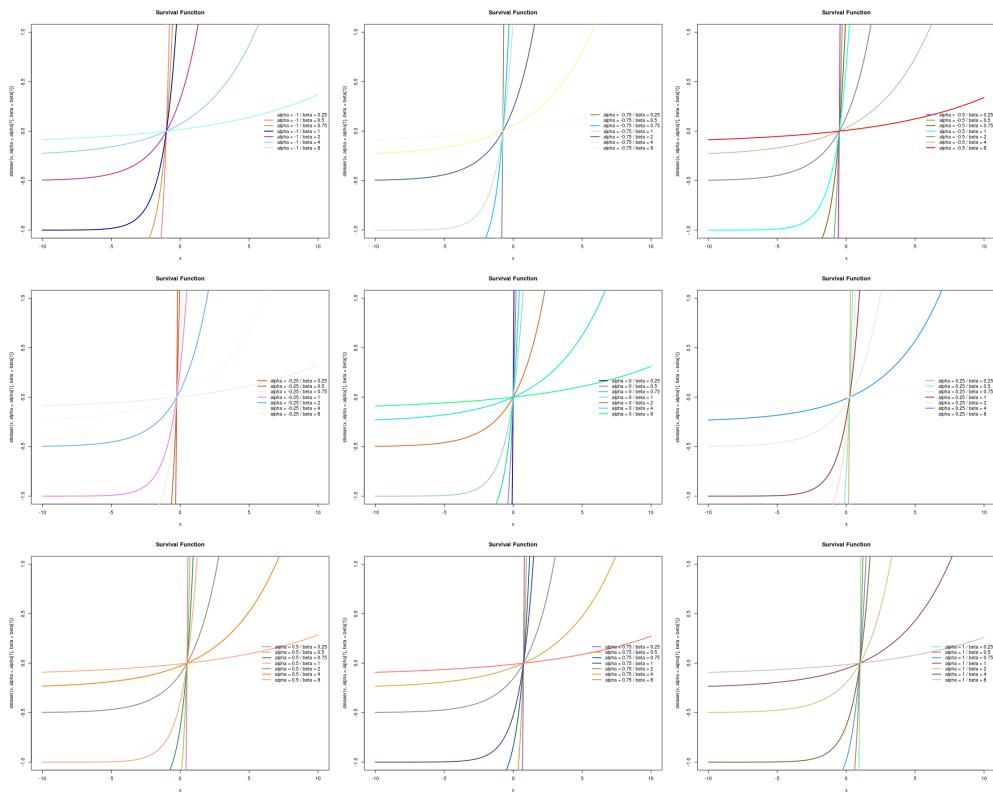


Figure 131: Teisser Distribution에 기반한 생존함수

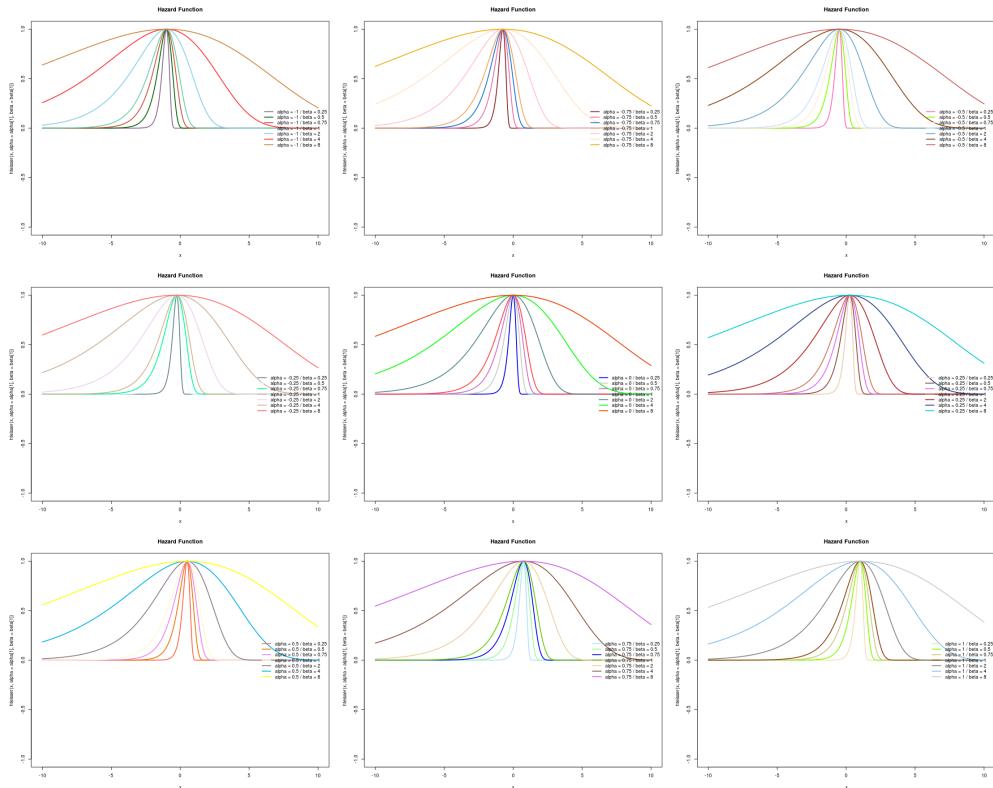


Figure 132: Teisser Distribution에 기반한 위험함수

Code 51. Teisser Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(pracma)
4
5
6 ##### teisser Distribution
7 ### parameter
8 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
9 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(-10, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dteisser = function(x, alpha = 0, beta = 1)
17 {
18   temp = (x-alpha)/beta
19   fx = (1/beta) * (exp(temp) - 1) * exp(1 + temp - exp(temp))
20   return(fx)
21 }
22
23
24 ### 누적분포함수
25 pteisser = function(x, alpha = 0, beta = 1)
26 {
27   fx = -(steisser(x, alpha = alpha, beta = beta) - 1)
28   return(fx)
29 }
30
31
32 ### 생존함수
33 steisser = function (x, alpha = 0, beta = 1)
34 {
35   temp = (x - alpha)/beta
36   fx = (1/beta) * (exp(temp) - 1)
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hteisser = function (x, alpha = 0, beta = 1)
43 {
44   fx = dteisser(x, alpha, beta) / steisser(x, alpha, beta)
45   return(fx)
46 }
47
48
49
50
51 ###### Plot
52 plot.teisser_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dteisser")
53 {
54   color=colorPalette(300)
55
56   len_alpha = length(alpha) # alpha 파라미터의 길이
57   len_beta = length(beta) # beta 파라미터의 길이
58
59   color_counter = 1
60   for (i in 1:len_alpha) ### 파라미터: alpha
61   {
62     color_counter_init = color_counter
63     legend_name = NULL;
64
65     if (func=="dteisser") # 수명분포
66     {
67       plot(x, dteisser(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
68           Distribution Function")
69       for (j in 1:len_beta) ### 파라미터: beta
70     {
71       lines(x, dteisser(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
72       color_counter = color_counter + 1;
73       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
74     }
75   }
76   else if (func == "pteisser") # 누적분포함수
77   {
78     plot(x, pteisser(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Cumulative Distribution Function")
79     for (j in 1:len_beta) ### 파라미터: beta
80     {
81       lines(x, pteisser(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
82       color_counter = color_counter + 1;
83       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
84     }
85   }
86   else if (func == "steisser") # 생존함수

```

```

87     {
88         plot(x, steisser(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival Function")
89         for (j in 1:len_beta) ### 파라미터: beta
90         {
91             lines(x, steisser(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
92             color_counter = color_counter + 1;
93             legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
94         }
95     }
96     else if (func == "hteisser") # 위험함수
97     {
98         plot(x, hteisser(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard Function")
99         for (j in 1:len_beta) ### 파라미터: beta
100        {
101            lines(x, hteisser(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
102            color_counter = color_counter + 1;
103            legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
104        }
105    }
106    legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
107 }
108 }
109 par(mfrow = c(3, 3))
110 plot.teisser_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dteisser")
112
113 par(mfrow = c(3, 3))
114 plot.teisser_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 2), func="pteisser")
115
116 par(mfrow = c(3, 3))
117 plot.teisser_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 1), func="steisser")
118
119 par(mfrow = c(3, 3))
120 plot.teisser_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 1), func="hteisser")

```

14.35 Triangular Distribution(Continuous)

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\begin{cases} \frac{2(t-a)}{cb^2} & \text{for } a \leq t \leq a + cb \\ \frac{2(a+b-x)}{(1-c)b^2} & \text{for } a + cb \leq t \leq a + b \end{cases}$	
생존함수 (신뢰도함수)	$\begin{aligned} S(t) &= P(T > t) \\ &= 1 - P(T \leq t) \\ &= 1 - F(t) \\ &= e^{-H(t)} \end{aligned}$	$\begin{cases} 1 - \frac{(t-a)^2}{cb^2} & \text{for } a \leq t \leq a + cb \\ \frac{(a+b-x)^2}{(1-c)b^2} & \text{for } a + cb \leq t \leq a + b \end{cases}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\begin{cases} \frac{2(t-a)}{cb^2 - (t-a)^2} & \text{for } a \leq t \leq a + cb \\ \frac{2}{a+b-x} & \text{for } a + cb \leq t \leq a + b \end{cases}$	IHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, 0 < c < 1$	

Table 64: Triangular 분포함수에 기반한 척도 함수

Triangular distribution은 c 파라메터에 따라 다음과 같은 분포가 된다.

- $c = 0.5$: Symmetric Triangular Distribution
- $c = 0$: Right-angled and Positively Skewed Triangular Distribution
- $c = 1$: Right-angled and Negatively Skewed Triangular Distribution

14.36 Uniform Distribution(Continuous)

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \frac{t-a}{b}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{a+b-t}$	IHR
변수	$a \leq t \leq a + b$		
파라메터	$a \in \mathbf{R}, b > 0$		

Table 65: Uniform 분포함수에 기반한 척도 함수

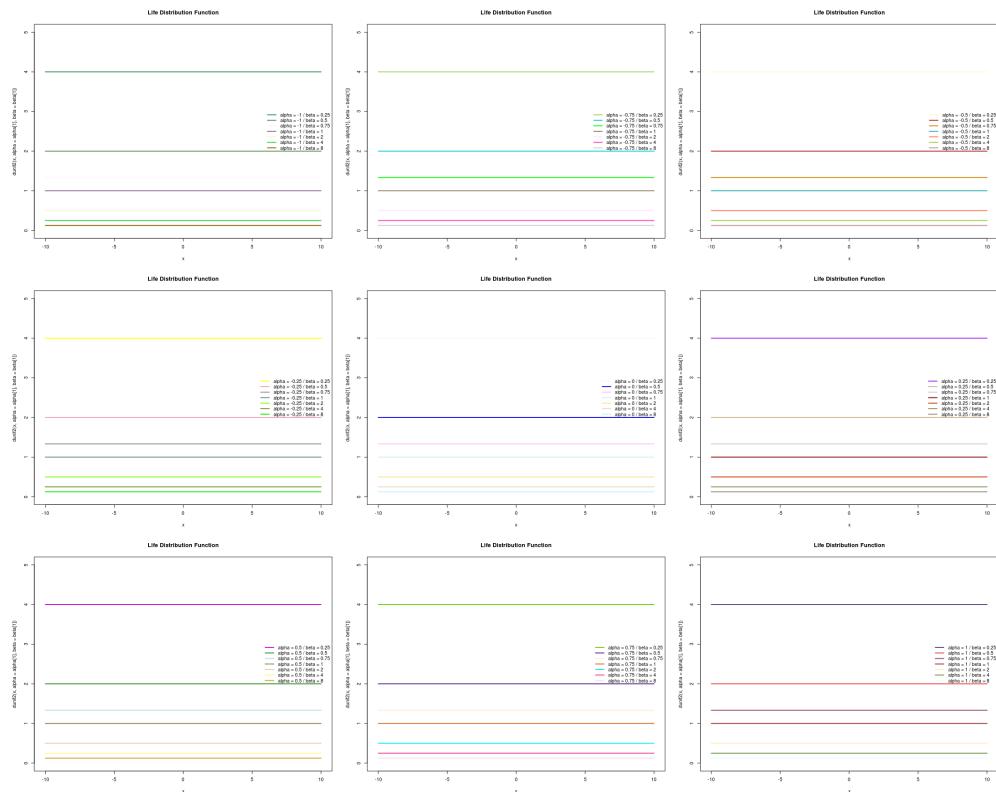


Figure 133: Uniform Distribution에 기반한 수명분포함수

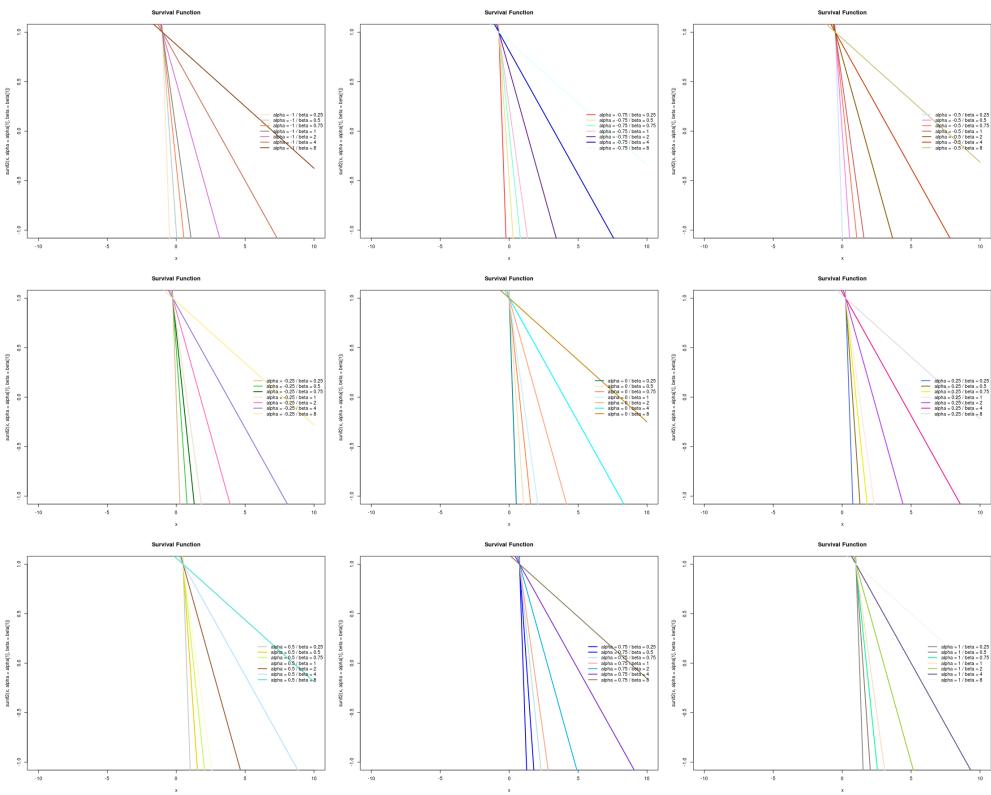


Figure 134: Uniform Distribution에 기반한 생존함수

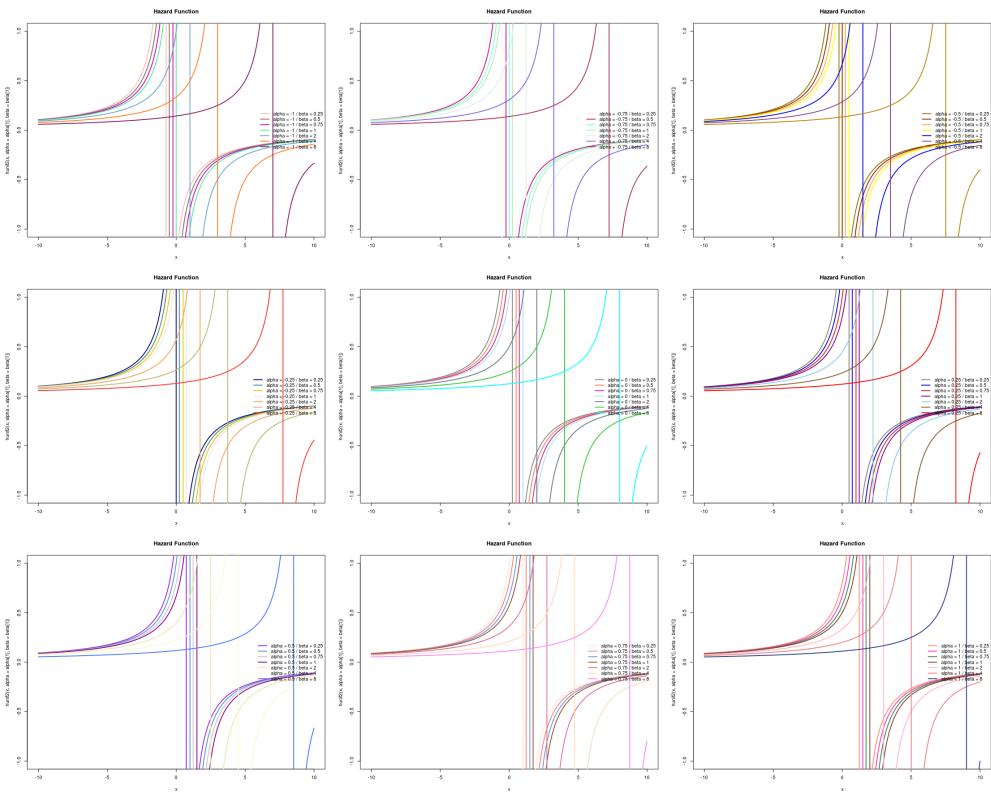


Figure 135: Uniform Distribution에 기반한 위험함수

Code 52. Uniform Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### unif2 Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(-10, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dunif2 = function(x, alpha = 0, beta = 1)
16 {
17   fx = rep(1/beta, length(x))
18   return(fx)
19 }
20
21
22 ### 누적분포함수
23 punif2 = function(x, alpha = 0, beta = 1)
24 {
25   fx = (x - alpha)/beta
26   return(fx)
27 }
28
29
30 ### 생존함수
31 sunif2 = function(x, alpha = 0, beta = 1)
32 {
33   fx = 1 - (x - alpha)/beta
34   return(fx)
35 }
36
37
38 ### 위험함수
39 hunif2 = function(x, alpha = 0, beta = 1)
40 {
41   fx = dunif2(x, alpha, beta) / sunif2(x, alpha, beta)
42   return(fx)
43 }
44
45
46
47
48
49 ##### Plot
50 plot.unif2_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dunif2")
51 {
52   color=colorPalette(300)
53
54   len_alpha = length(alpha) # alpha 파라미터의 길이
55   len_beta = length(beta) # beta 파라미터의 길이
56
57   color_counter = 1
58   for (i in 1:len_alpha) ### 파라미터: alpha
59   {
60     color_counter_init = color_counter
61     legend_name = NULL;
62
63     if (func=="dunif2") # 수명분포
64     {
65       plot(x, dunif2(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
66           Distribution Function")
67       for (j in 1:len_beta) ### 파라미터: beta
68       {
69         lines(x, dunif2(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
70         color_counter = color_counter + 1;
71         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
72       }
73     }
74     else if (func == "punif2") # 누적분포함수
75     {
76       plot(x, punif2(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
77           Cumulative Distribution Function")
78       for (j in 1:len_beta) ### 파라미터: beta
79       {
80         lines(x, punif2(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
81         color_counter = color_counter + 1;
82         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
83     }
84     else if (func == "sunif2") # 생존함수
85     {
86       plot(x, sunif2(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Survival
87           Function")
88     }
89   }
90 }
```

```

86     for (j in 1:len_beta) ### 파라메터: beta
87     {
88       lines(x, sunif2(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
89       color_counter = color_counter + 1;
90       legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
91     }
92   }
93 else if (func == "hunif2") # 위험함수
94 {
95   plot(x, hunif2(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Hazard
96   Function")
97   for (j in 1:len_beta) ### 파라메터: beta
98   {
99     lines(x, hunif2(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
100    color_counter = color_counter + 1;
101    legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
102  }
103  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
104 }
105 }
106
107 par(mfrow = c(3, 3))
108 plot.unif2_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dunif2")
109
110 par(mfrow = c(3, 3))
111 plot.unif2_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 2), func="punif2")
112
113 par(mfrow = c(3, 3))
114 plot.unif2_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 1), func="sunif2")
115
116 par(mfrow = c(3, 3))
117 plot.unif2_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(-1, 1), func="hunif2")

```

14.37 V-shaped Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\begin{cases} \frac{2(2a+b-2t)}{b^2} & \text{for } a \leq t \leq a + \frac{b}{2} \\ \frac{2(2t-2a-b)}{b^2} & \text{for } a + \frac{b}{2} \leq t \leq a + b \end{cases}$	
생존함수 (신뢰도함수)	$\begin{aligned} S(t) &= P(T > t) \\ &= 1 - P(T \leq t) \\ &= 1 - F(t) \\ &= e^{-H(t)} \end{aligned}$	$\begin{cases} 1 - \frac{2(t-a)(a+b-t)}{b^2} & \text{for } a \leq t \leq a + \frac{b}{2} \\ 0.5 - \frac{(2t-2a-b)^2}{b^2} & \text{for } a + \frac{b}{2} \leq t \leq a + b \end{cases}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\begin{cases} \frac{2(2a+b-2t)}{b^2 + 2(a-t)(a+b-t)} & \text{for } a \leq t \leq a + \frac{b}{2} \\ \frac{2a+b-2t}{(a-t)(a+b-t)} & \text{for } a + \frac{b}{2} \leq t \leq a + b \end{cases}$	DIHR
변수 파라메터		$\begin{aligned} t &\geq a \\ a &\in \mathbf{R}, b > 0 \end{aligned}$	

Table 66: V-shaped 분포함수에 기반한 척도 함수

14.38 Wald Distribution

이 분포는 inverse normal(Gaussian) distribution이라고도 알려져 있다. 자세한 것은 Chapter 14.27.4을 참고하도록 하자.

14.39 Weibull Distribution(와이블 분포)

지수분포는 무기역성 때문에 응용 분야가 제한될 뿐만 아니라, 위험률(고장률)이 나이에 관계없이 일정하다는 성질이 비현실적이며, 수명자료의 분석에 부적합한 경우도 종종 있다.

와이블 분포는 위험률이 상수인 경우, 증가하는 경우, 감소하는 경우 등을 모형화하는 수명분포로, 다양한 형태의 수명자료를 분석하는 데 널리 활용되고 있다.

14.39.1 Weibull Distribution with 2 Parameters(2-모수 와이블 분포)

Table 67: 2모수 와이블 분포함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$\frac{\alpha}{\theta^\alpha} t^{\alpha-1} e^{-(\frac{t}{\theta})^\alpha}$
생존함수 (신뢰도함수)	$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) = e^{-H(t)}$	$e^{-(\frac{t}{\theta})^\alpha}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{\alpha}{\theta^\alpha} t^{\alpha-1}$
변수		$t \geq 0$
파라미터	$\alpha > 0$ (형상 모수; shape parameter), $\theta > 0$ (척도 모수; scale parameter)	

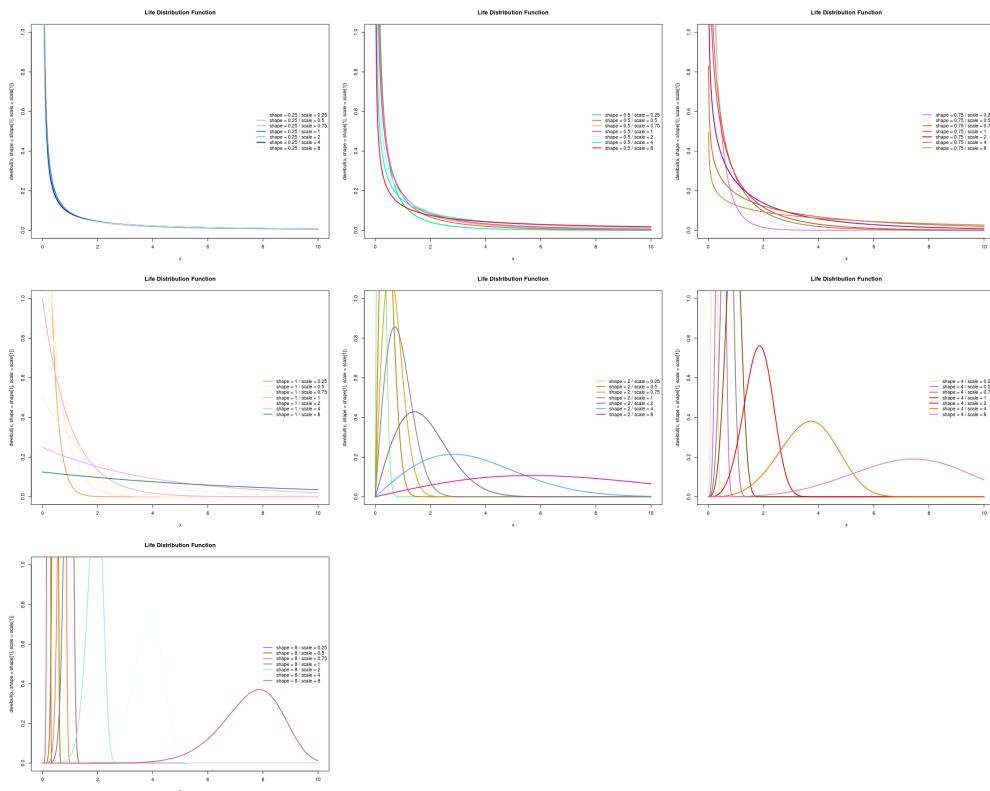


Figure 136: 2모수 와이블 Distribution에 기반한 수명분포함수

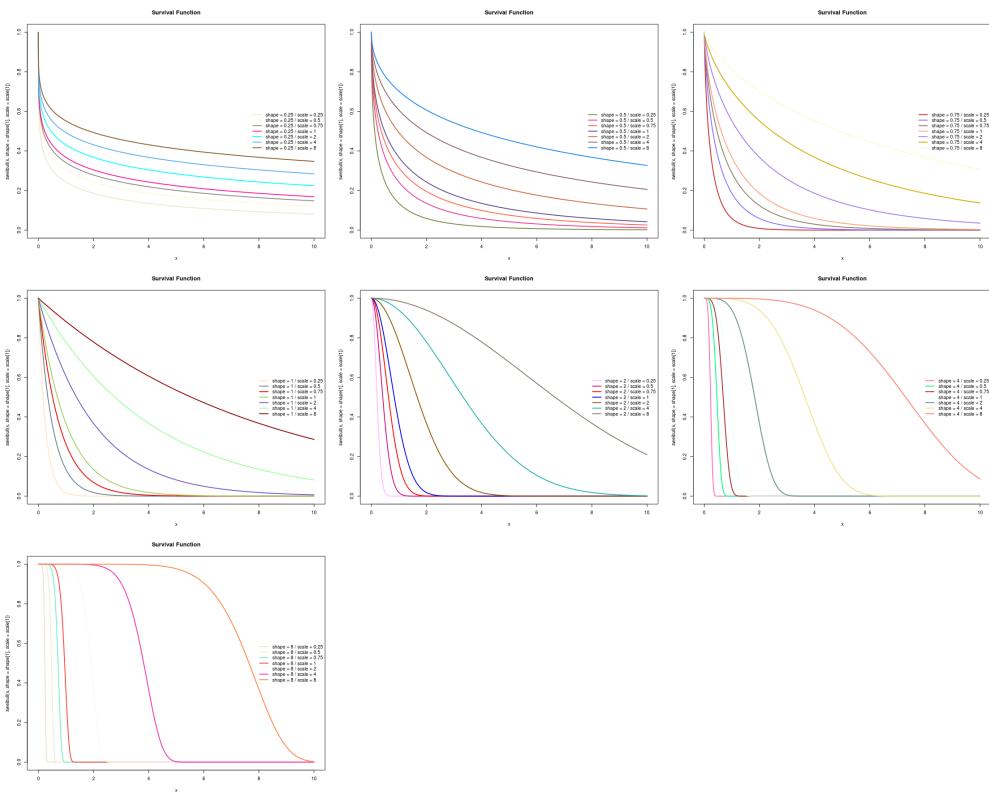


Figure 137: 2모수 와이블 Distribution에 기반한 생존함수

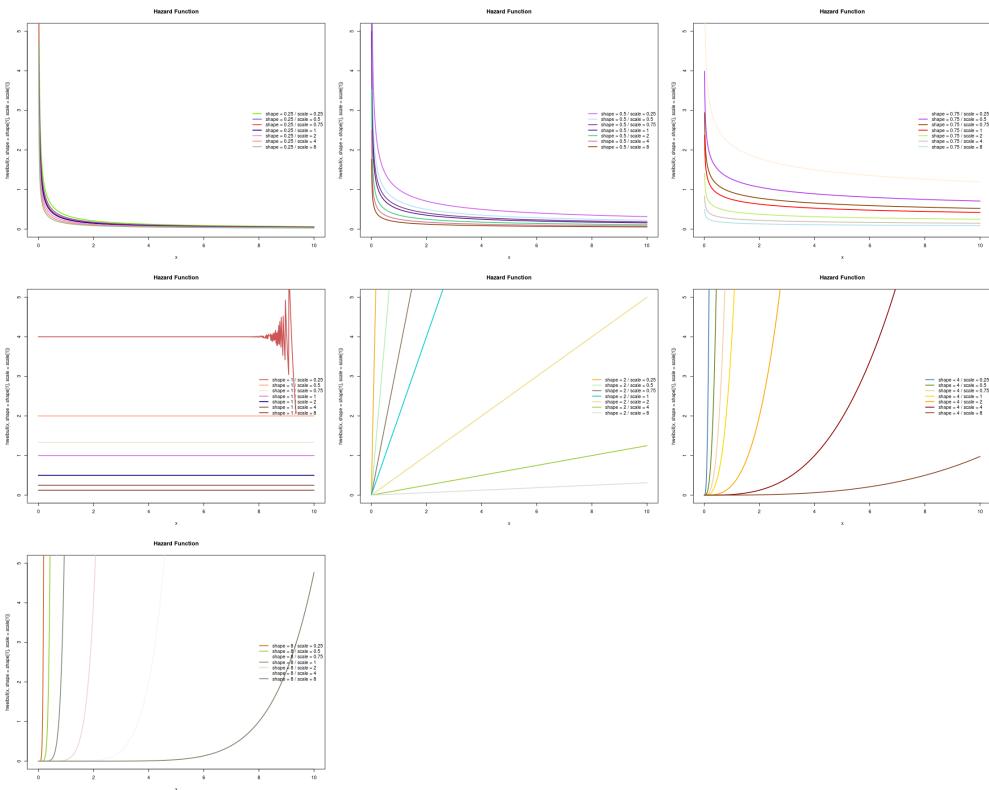


Figure 138: 2모수 와이블 Distribution에 기반한 위험함수

Code 53. 2모수 와이블 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4 ###### 극치 분포: Gumbel 최대값 분포
5 #### parameter
6 shape = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
7 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8
9 #### input varialbe
10 x = seq(0, 10, length.out = 1000)
11
12
13 #### 수명 분포
14 dweibull(x, shape = 1, scale = 1)
15
16
17 #### 분위수 함수
18 qweibull(x, shape = 1, scale = 1)
19
20
21 #### 난수 함수
22 rweibull(x, shape = 1, scale = 1)
23
24
25 #### 누적분포함수
26 pweibull(x, shape = 1, scale = 1)
27
28
29 #### 생존함수
30 sweibull = function (x, shape = 1, scale = 1)
31 {
32   fx = 1 - pweibull(x, shape = shape, scale = scale)
33   return(fx)
34 }
35
36
37 #### 위험함수
38 hweibull = function (x, shape = 1, scale = 0)
39 {
40   fx = dweibull(x, shape = shape, scale = scale) / sweibull(x, shape = shape, scale = scale)
41   return(fx)
42 }
43
44
45
46 #### 위험함수
47 hweibull = function (x, scale = 1, shape = 0)
48 {
49   fx = dweibull(x, shape = shape, scale = scale) / sweibull(x, shape = shape, scale = scale)
50   return(fx)
51 }
52
53
54
55
56
57 ###### Plot
58 plot.weibull_seq = function(x, shape = 1, scale = 0, xlim=c(0, 10), ylim=c(0, 5), func="dweibull")
59 {
60   color=colorPalette(300)
61
62   len_shape = length(shape) # shape 파라메터의 길이
63   len_scale = length(scale) # scale 파라메터의 길이
64
65   color_counter = 1
66   for (i in 1:len_shape) ### 파라메터: shape
67   {
68     color_counter_init = color_counter
69     legend_name = NULL;
70
71     if (func=="dweibull") # 수명분포
72     {
73       plot(x, dweibull(x, shape=shape[i], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
74           Distribution Function")
75       for (j in 1:len_scale) ### 파라메터: scale
76       {
77         lines(x, dweibull(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
78         color_counter = color_counter + 1;
79         legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
80       }
81     }
82     else if (func == "pweibull") # 누적분포함수
83     {
84       plot(x, pweibull(x, shape=shape[i], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
85           Cumulative Distribution Function")
86       for (j in 1:len_scale) ### 파라메터: scale
87       {

```

```

87     lines(x, pweibull(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
88     color_counter = color_counter + 1;
89     legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
90   }
91 }
92 else if (func == "sweibull") # 생존함수
93 {
94   plot(x, sweibull(x, shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
95       "Survival Function")
96   for (j in 1:len_scale) ### 파라미터: scale
97   {
98     lines(x, sweibull(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
99     color_counter = color_counter + 1;
100    legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
101  }
102 else if (func == "hweibull") # 위험함수
103 {
104   plot(x, hweibull(x, shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
105       "Hazard Function")
106   for (j in 1:len_scale) ### 파라미터: scale
107   {
108     lines(x, hweibull(x, shape=shape[i], scale=scale[j]), col=color[color_counter], lwd=2);
109     color_counter = color_counter + 1;
110     legend_name = c(legend_name, paste("shape = ", shape[i], " / scale = ", scale[j], sep=""))
111   }
112   legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
113 }
114 }
115 par(mfrow = c(3, 3))
116 plot.weibull_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="dweibull")
117
118 par(mfrow = c(3, 3))
119 plot.weibull_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="pweibull")
120
121 par(mfrow = c(3, 3))
122 plot.weibull_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 1), func="sweibull")
123
124 par(mfrow = c(3, 3))
125 plot.weibull_seq(x, shape, scale, xlim=c(min(x), max(x)), ylim=c(0, 5), func="hweibull")
126

```

척도 모수(scale parameter)는 종종 $\lambda = \frac{1}{\theta}$ 로 표시되는 경우도 있다.

Figure ??를 통해, 형상 모수 α 에 따라 $h(t)$ 의 형태가 바뀌는 모습을 확인할 수 있다.

- $\alpha = 1$: 위험률(고장률) $h(t)$ 가 상수
- $\alpha > 1$: 위험률(고장률) $h(t)$ 가 증가 함수
- $\alpha < 1$: 위험률(고장률) $h(t)$ 가 감소 함수

Note 22. 2모수 와이블분포에 기반한 평균잔여수명함수

$$m(t) = \frac{\int_t^\infty S(u)du}{S(t)} = \frac{\int_t^\infty e^{-(\frac{u}{\theta})^\alpha} du}{e^{-(\frac{t}{\theta})^\alpha}}$$

이 때,

- $\alpha = 1$ 인 경우 $m(t) = 0$
- $\alpha = 2$ 인 경우

$$\begin{aligned} \int_t^\infty e^{-(\frac{u}{\theta})^2} du &= \sqrt{\pi}\theta \int_t^\infty \frac{1}{\sqrt{\pi}\theta} e^{-\frac{u^2}{\theta^2}} du \\ &= \sqrt{\pi}\theta [1 - \Phi(t)] \end{aligned}$$

이므로,

$$m(t) = e^{-(\frac{t}{\theta})^2} \sqrt{\pi}\theta [1 - \Phi(t)]$$

여기서, $\Phi(t) = P(Z \leq t)$: 표준정규분포의 누적분포함수

그러나 일반적인 α 의 값에 대한 $m(t)$ 의 공식은 계산할 수 없으며, 수치적인 방법을 이용하여야 한다.

Note 23. 2모수 와이블분포에 기반한 k 차 적률

$$\begin{aligned} E(T^k) &= \int_0^\infty t^k f(t) dt \\ &= \int_0^\infty t^k \frac{\alpha}{\theta^\alpha} t^{\alpha-1} e^{-(\frac{t}{\theta})^\alpha} dt \\ &= \int_0^\infty t^k \frac{\alpha}{\theta} \left(\frac{t}{\theta}\right)^{\alpha-1} e^{-(\frac{t}{\theta})^\alpha} dt \\ &= \int_0^\infty t^{k+\alpha-1} \frac{1}{\theta^\alpha} \alpha e^{-(\frac{t}{\theta})^\alpha} dt \\ &= \int_0^\infty \left(y^{\frac{1}{\alpha}}\right)^{k+\alpha-1} \frac{1}{\theta^\alpha} \alpha e^{-\frac{y}{\theta^\alpha}} \frac{1}{\alpha \left(y^{\frac{1}{\alpha}}\right)^{\alpha-1}} dy \quad (\text{substituting } t^\alpha = y) \\ &= \frac{1}{\theta^\alpha} \Gamma\left(1 + \frac{k}{\alpha}\right) \theta^{\alpha(1+\frac{k}{\alpha})} \\ &= \theta^k \Gamma\left(1 + \frac{k}{\alpha}\right) \end{aligned}$$

Note 24. 100p 백분위수

$$\begin{aligned} S(t_p) &= e^{-\left(\frac{t_p}{\theta}\right)^\alpha} \\ &= 1 - p \\ \Rightarrow \left(\frac{t_p}{\theta}\right)^\alpha &= -\ln(1 - p) \\ \Rightarrow t_p &= \theta [-\ln(1 - p)]^{\frac{1}{\alpha}} \end{aligned}$$

와이블 분포의 기본적인 성질은 다음과 같다.

1. 만일 T_1, T_2, \dots, T_n 이 서로 독립이고, 각각 형상 모수 α 와 척도 모수 θ 가 동일한 와이블 분포를 가진다고 하자.

이 경우, $T = \min(T_1, T_2, \dots, T_n) \sim Weibull(\alpha, \theta n^{-\frac{1}{\alpha}})$ 가 된다.

Proof: T 의 생존함수 $S(t)$ 가 다음과 같이 계산된다.

$$\begin{aligned} S(t) &= P[\min(T_1, T_2, \dots, T_n) > t] \\ &= P[T_1 > t, T_2 > t, \dots, T_n > t] \\ &= P[T_1 > t]P[T_2 > t] \cdots P[T_n > t] \\ &= e^{-(\frac{t}{\theta})^\alpha} e^{-(\frac{t}{\theta})^\alpha} \cdots e^{-(\frac{t}{\theta})^\alpha} \\ &= e^{-n(\frac{t}{\theta})^\alpha} \\ &= e^{-\left(\frac{n^{\frac{1}{\alpha}} t}{\theta}\right)^\alpha} \sim Weibull(\alpha, \theta n^{-\frac{1}{\alpha}}) \end{aligned}$$

Note 25. 2모수 와이블분포를 따르는 완전자료에 대한 모수 추정

- 점추정

시험 중인 n 개의 부품(환자)에 대한 고장(재발)시간 t_1, t_2, \dots, t_n 이 관측되었다고 가정하고, 이 완전자료를 이용하면, 2모수 와이블분포의 형상모수 α 와 척도모수 θ 의 최대가능도추정량(MLE)은 다음과 같이 계산된다.

$$\begin{aligned} L(t; \alpha, \theta) &= \prod_{i=1}^n \frac{\alpha}{\theta^\alpha} t_i^{\alpha-1} e^{-\sum_{i=1}^n \left(\frac{t_i}{\theta}\right)^\alpha} \\ &= \left(\frac{\alpha}{\theta^\alpha}\right)^n \prod_{i=1}^n t_i^{\alpha-1} e^{-\sum_{i=1}^n \left(\frac{t_i}{\theta}\right)^\alpha} \end{aligned}$$

양변에 자연로그를 취하면

$$\ln L(t; \alpha, \theta) = n(\ln \alpha - \alpha \ln \theta) + (\alpha - 1) \sum_{i=1}^n t_i - \sum_{i=1}^n \left(\frac{t_i}{\theta}\right)^\alpha \quad (8)$$

식 (8)에 대해 α 와 θ 에 대한 1차도함수를 구하고 0으로 놓으면, 다음과 같은 두 개의 방정식을 얻는다.

$$\frac{n}{\alpha} - nl\theta + \sum_{i=1}^n lt_i - \sum_{i=1}^n \left(\frac{t_i}{\theta}\right)^\alpha l\left(\frac{t_i}{\theta}\right) = 0 \quad (9)$$

$$\frac{-n\alpha}{\theta} + \frac{\alpha}{\theta^{\alpha+1}} \sum_{i=1}^n t_i^\alpha = 0 \quad (10)$$

식 (9), (10)을 동시에 만족시키는 α 와 θ 의 값이 MLE가 된다.

우선, 식 (10)를 θ 에 대하여 풀면

$$\hat{\theta} = \left[\frac{\sum_{i=1}^n t_i^\alpha}{n} \right]^{\frac{1}{\alpha}}$$

이 $\hat{\theta}$ 를 식 (9)에 대입하면, 다음과 같은 α 에 대한 방정식을 얻는다.

$$\frac{1}{\alpha} + \frac{1}{n} \sum_{i=1}^n \ln t_i - \frac{\sum_{i=1}^n t_i^\alpha \ln t_i}{\sum_{i=1}^n t_i^\alpha} = 0 \quad (11)$$

식 (11)에 대한 해 $\hat{\alpha}$ 가 MLE이며, Newton-Raphson 방법을 이용하여 수치적으로 계산된다.
일단 $\hat{\alpha}$ 가 결정되면, θ 의 MLE $\hat{\theta}$ 는 다음과 같다.

$$\hat{\theta} = \left[\frac{\sum_{i=1}^n t_i^{\hat{\alpha}}}{n} \right]^{\frac{1}{\hat{\alpha}}}$$

- 구간 추정

$\hat{\alpha}$ 와 $\hat{\theta}$ 의 정확한 분포를 모르는 경우, α 와 θ 에 대한 소표본 신뢰구간을 구할 수 없다.

그러나 대표본인 경우에는 α 와 θ 에 대한 $100(1-\nu)\%$ 신뢰구간이 근사적으로 계산될 수 있다.
우선, 다음과 같이 세팅을 하자.

$$S_0 = \sum_{i=1}^n t_i^{\hat{\alpha}} \quad S_1 = \sum_{i=1}^n t_i^{\hat{\alpha}} (\ln t_i) \quad S_2 = \sum_{i=1}^n t_i^{\hat{\alpha}} (\ln t_i)^2$$

그러면, $\hat{\alpha}$, $\hat{\theta}^{\hat{\alpha}}$ 에 대한 분산은 각각 다음과 같이 근사적으로 계산될 수 있다.

$$\begin{aligned} Var(\hat{\alpha}) &= \frac{\hat{\alpha}^2 S_0^2}{n \left(S_0^2 + \hat{\alpha}^2 S_0 S_2 - \hat{\alpha}^2 S_1^2 \right)} \\ Var(\hat{\theta}^{\hat{\alpha}}) &= \frac{S_0}{n^2} \left(\frac{S_0}{\hat{\alpha}^2} + S_2 \right) Var(\hat{\alpha}) \end{aligned}$$

따라서, α 에 대한 신뢰구간은 다음과 같다.

$$\left[\hat{\alpha} - z_{\frac{\nu}{2}} \sqrt{Var(\hat{\alpha})} < \alpha < \hat{\alpha} + z_{\frac{\nu}{2}} \sqrt{Var(\hat{\alpha})} \right]$$

또한, $\theta^{\hat{\alpha}}$ 에 대한 신뢰구간은 다음과 같다.

$$\left[\hat{\theta}^{\hat{\alpha}} - z_{\frac{\nu}{2}} \sqrt{Var(\hat{\theta}^{\hat{\alpha}})} < \theta^{\hat{\alpha}} < \hat{\theta}^{\hat{\alpha}} + z_{\frac{\nu}{2}} \sqrt{Var(\hat{\theta}^{\hat{\alpha}})} \right]$$

θ 에 대한 신뢰구간은 $\theta = (\theta^{\hat{\alpha}})^{\frac{1}{\hat{\alpha}}}$ 의 관계를 이용하여 구한다.

Note 26. 2모수 와이블분포를 따르는 Type II 우중도절단자료(정수중단자료)에 대한 모수 추정

2모수 와이블 수명분포를 따르는 n 개의 부품(환자)을 가지고 수명시험을 수행하여 $r(\leq n)$ 개의 고장(재발)이 발생한 시점에서 시험을 중단하였을 때, 얻어진 고장(재발)자료를 크기 순서대로 나열하여 $t_{(1)} \leq t_{(2)} \leq \dots \leq t_{(r)}$ 으로 나타낸다. 나머지 $(n - r)$ 개의 부품(환자)에 대한 고장(재발) 자료는 시점 $t_{(r)}$ 에서 결단된다.

Type II 우중도절단자료(정수중단자료)가 주어진 경우의 α 와 θ 에 대한 가능도 함수는 다음과 같다.

$$\begin{aligned} L(t; \theta) &= \frac{n!}{(n-r)!} \left[\prod_{i=1}^r \frac{\alpha}{\theta} \left(\frac{t_{(i)}}{\theta} \right)^{\alpha-1} e^{-\left(\frac{t_{(i)}}{\theta} \right)^\alpha} \right] e^{-\left(\frac{t_{(r)}}{\theta} \right)^\alpha} \\ &= \frac{n!}{(n-r)!} \left(\frac{\alpha}{\theta} \right)^r \prod_{i=1}^r \left(\frac{t_{(i)}}{\theta} \right)^{\alpha-1} e^{-\sum_{i=1}^r \left(\frac{t_{(i)}}{\theta} \right)^\alpha - (n-r) \left(\frac{t_{(r)}}{\theta} \right)^\alpha} \end{aligned}$$

양변에 자연로그를 취하면

$$\ln L(t; \alpha, \theta) = n(\ln \alpha - \alpha \ln \theta) + (\alpha - 1) \sum_{i=1}^r t_i - \sum_{i=1}^r \left(\frac{t_i}{\theta} \right)^\alpha \quad (12)$$

식 (12)에 대해 α 와 θ 에 대한 1차도함수를 구하고 0으로 놓으면, 다음과 같은 두 개의 방정식을 얻는다.

$$-r + \frac{1}{\theta^\alpha} \sum_{i=1}^r t_{(i)}^\alpha + (n-r)t_{(r)}^\alpha = 0 \quad (13)$$

$$\frac{r}{\alpha} + \sum_{i=1}^r t_{(i)}^\alpha - \frac{1}{\theta^\alpha} \sum_{i=1}^r t_{(i)}^\alpha \ln t_{(i)} = (n-r)t_{(i)}^\alpha \ln t_{(r)} = 0 \quad (14)$$

식 (13)로부터 얻어진 θ^α 의 값을 식 (14)에 대입하면

$$\frac{\sum_{i=1}^r t_{(i)}^\alpha \ln t_{(i)} + (n-r)t_{(r)}^\alpha \ln t_{(r)}}{\sum_{i=1}^r t_{(i)}^\alpha + (n-r)t_{(r)}^\alpha} - \frac{1}{r} \sum_{i=1}^r t_{(i)} - \frac{1}{\alpha} = 0 \quad (15)$$

식 (15)을 만족하는 α 의 값이 MLE $\hat{\alpha}$ 가 된다. 이 $\hat{\alpha}$ 의 값은 Newton-Raphson 방법에 의해 수치적으로 찾아질 수 있으며, 일단 $\hat{\alpha}$ 의 값이 결정되면 θ 의 MLE $\hat{\theta}$ 는 다음과 같다.

$$\hat{\theta} = \left[\frac{\sum_{i=1}^r t_{(i)}^{\hat{\alpha}} + (n-r)t_{(r)}^{\hat{\alpha}}}{r} \right]^{\frac{1}{\hat{\alpha}}}$$

참고로 $\hat{\alpha}$, $\hat{\theta}$ 의 분산은 시뮬레이션으로 구할 수 있다.[12]

14.39.2 Weibull Distribution with 3 Parameters(3-모수 와이블 분포)

Table 68: 3모수 와이블 분포함수에 기반한 척도 함수

척도 함수	기호	내용
수명분포함수	$f(t)$	$\frac{\alpha}{\theta} \left(\frac{t-\gamma}{\theta}\right)^{\alpha-1} e^{-\left(\frac{t-\gamma}{\theta}\right)^\alpha}$
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{-\left(\frac{t-\gamma}{\theta}\right)^\alpha}$
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{\alpha}{\theta^\alpha} (t - \gamma)^{\alpha-1}$
변수		$t \geq 0$
파라메터		$\alpha > 0$ (형상 모수; shape parameter), $\theta > 0$ (척도 모수; scale parameter) $0 < \gamma < \infty$ (위치 모수; location parameter), $t \geq \gamma$

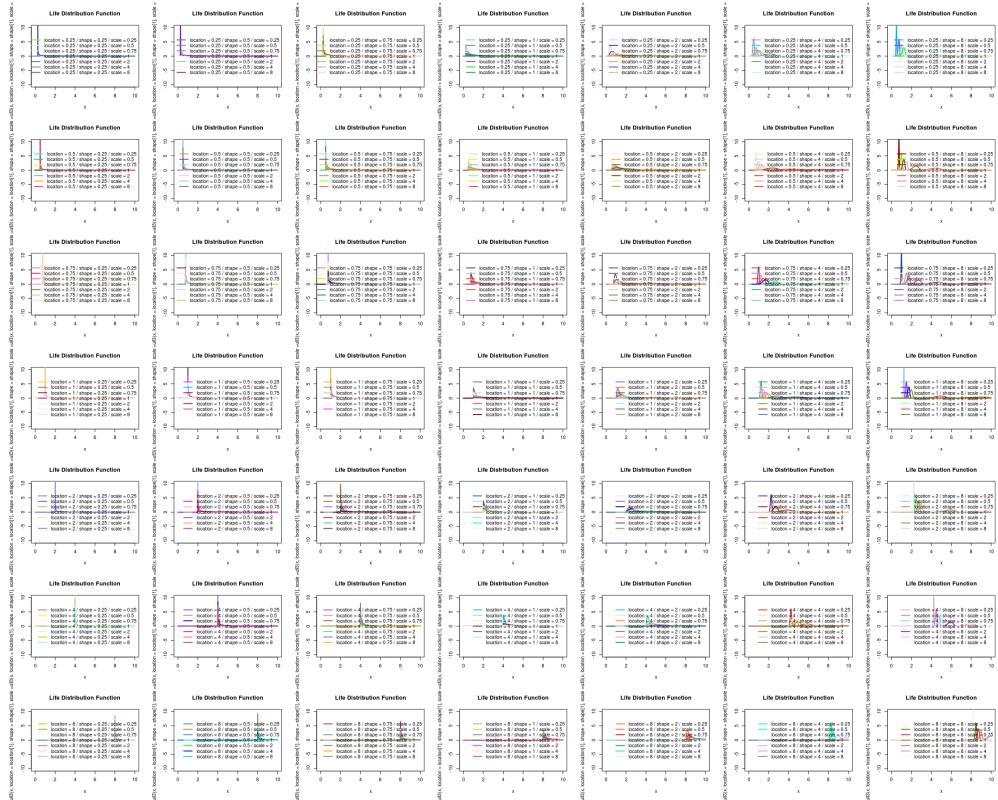


Figure 139: 3모수 와이블 Distribution에 기반한 수명분포함수



Figure 140: 3모수 와이블 Distribution에 기반한 생존함수

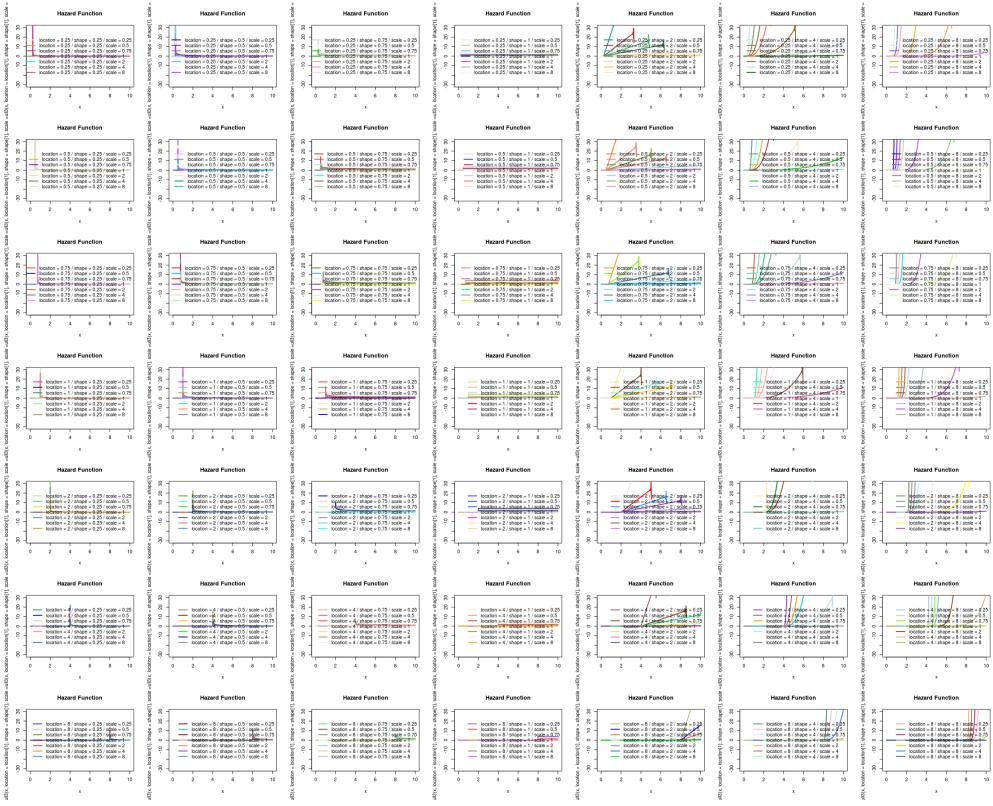


Figure 141: 3모수 와이블 Distribution에 기반한 위험함수

Code 54. 3모수 와이블 Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### weibull Distribution with Location Parameters
6 ### parameter
7 shape = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
8 scale = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 location = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(0, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dweibull3 = function(x, shape=shape, scale=scale, location=location)
17 {
18   fx = dweibull(x-location, shape=shape, scale=scale)
19   return(fx)
20 }
21
22
23 ### 분위수 함수
24 qweibull3 = function(x, shape=shape, scale=scale, location=location)
25 {
26   fx = qweibull(x-location, shape=shape, scale=scale)
27   return(fx)
28 }
29
30
31 ### 난수 함수
32 rweibull3 = function(x, shape=shape, scale=scale, location=location)
33 {
34   fx = rweibull(x-location, shape=shape, scale=scale)
35   return(fx)
36 }
37
38
39 ### 누적분포함수
40 pweibull3 = function(x, shape=shape, scale=scale, location=location)
41 {
42   fx = pweibull(x-location, shape=shape, scale=scale)
43   return(fx)
44 }
45
46
47 ### 생존함수
48 sweibull3 = function(x, shape=1, scale=1, location=0)
49 {
50   fx = 1 - pweibull3(x, shape=shape, scale=scale, location=location)
51   return(fx)
52 }
53
54
55 ### 위험함수
56 hweibull3 = function (x, shape=shape, scale=scale, location=0)
57 {
58   fx = dweibull3(x, shape=shape, scale=scale, location=location) / sweibull3(x, shape=shape, scale=scale, location=location)
59   return(fx)
60 }
61
62
63
64
65
66 ##### Plot
67 plot.weibull3_seq = function(x, shape = 1, scale = 1, location = 1, xlim=c(0, 10), ylim=c(0, 5), func="dweibull3")
68 {
69   color=colorPalette(300)
70
71   len_location = length(location) # location 파라메터의 길이
72   len_shape = length(shape) # shape 파라메터의 길이
73   len_scale = length(scale) # scale 파라메터의 길이
74
75   color_counter = 1
76   for (i in 1:len_location) ### 파라메터: location
77   {
78     if (func=="dweibull3") # 수명분포
79     {
80       for (j in 1:len_shape) ### 파라메터: shape
81       {
82         color_counter_init = color_counter
83         legend_name = NULL;
84         plot(x, dweibull3(x, location=location[i], shape=shape[j], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
85           =2, type = 'n', main="Life Distribution Function")
86         for (k in 1:len_scale) ### 파라메터: scale
87         {
88           lines(x, dweibull3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);

```

```

88         color_counter = color_counter + 1;
89         legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
90                         sep=""))
90     }
91     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
92   }
93 }
94 else if (func == "pweibull3") # 누적분포함수
95 {
96   for (j in 1:len_shape) ### 파라미터: shape
97   {
98     color_counter_init = color_counter
99     legend_name = NULL;
100    plot(x, pweibull3(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
101        =2, type = 'n', main="Cumulative Distribution Function")
102    for (k in 1:len_scale) ### 파라미터: scale
103    {
104      lines(x, pweibull3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
105      color_counter = color_counter + 1;
106      legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
107                      sep=""))
108    }
109    legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
110  }
111 else if (func == "sweibull3") # 생존함수
112 {
113   for (j in 1:len_shape) ### 파라미터: shape
114   {
115     color_counter_init = color_counter
116     legend_name = NULL;
117     plot(x, sweibull3(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
118        =2, type = 'n', main="Survival Function")
119     for (k in 1:len_scale) ### 파라미터: scale
120     {
121       lines(x, sweibull3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
122       color_counter = color_counter + 1;
123       legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
124                       sep=""))
125     }
126     legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
127   }
128 else if (func == "hweibull3") # 위험함수
129 {
130   for (j in 1:len_shape) ### 파라미터: shape
131   {
132     color_counter_init = color_counter
133     legend_name = NULL;
134     plot(x, hweibull3(x, location=location[1], shape=shape[1], scale=scale[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
135        =2, type = 'n', main="Hazard Function")
136     for (k in 1:len_scale) ### 파라미터: scale
137     {
138       lines(x, hweibull3(x, location=location[i], shape=shape[j], scale=scale[k]), col=color[color_counter], lwd=2);
139       color_counter = color_counter + 1;
140       legend_name = c(legend_name, paste("location = ", location[i], " / shape = ", shape[j], " / scale = ", scale[k],
141                       sep=""))
142     }
143   }
144 }
145 par(mfrow = c(7, 7))
146 plot.weibull3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dweibull3")
147
148 par(mfrow = c(7, 7))
149 plot.weibull3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pweibull3")
150
151 par(mfrow = c(7, 7))
152 plot.weibull3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sweibull3")
153
154 par(mfrow = c(7, 7))
155 plot.weibull3_seq(x, shape, scale, location, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hweibull3")

```

시스템의 수명이 일정기간 동안 보장되는 경우에는, 3모수 와이블분포를 적용하는 것이 보다 적절하다.

척도 모수(scale parameter)는 종종 $\lambda = \frac{1}{\theta}$ 로 표시되는 경우도 있다.
위치모수 $\gamma = 0$ 이면 2모수 와이블 분포가 된다.

14.39.3 Log-Weibull Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{1}{b} e^{\frac{t-a}{b}} - e^{\frac{t-a}{b}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$e^{-e^{\frac{t-a}{b}}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{1}{b} e^{\frac{t-a}{b}}$	IHR
변수		$t \geq 0$	
파라메터		$a \in \mathbf{R}, b > 0$	

Table 69: Log-Weibull 분포함수에 기반한 척도 함수

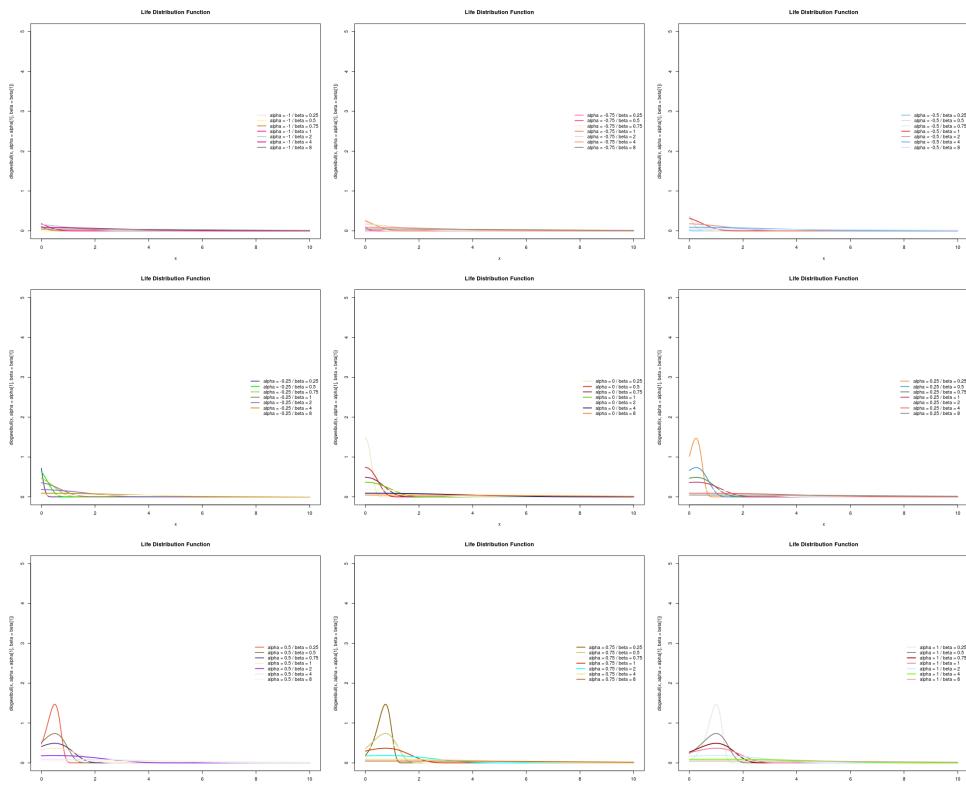


Figure 142: Log-Weibull Distribution에 기반한 수명분포함수

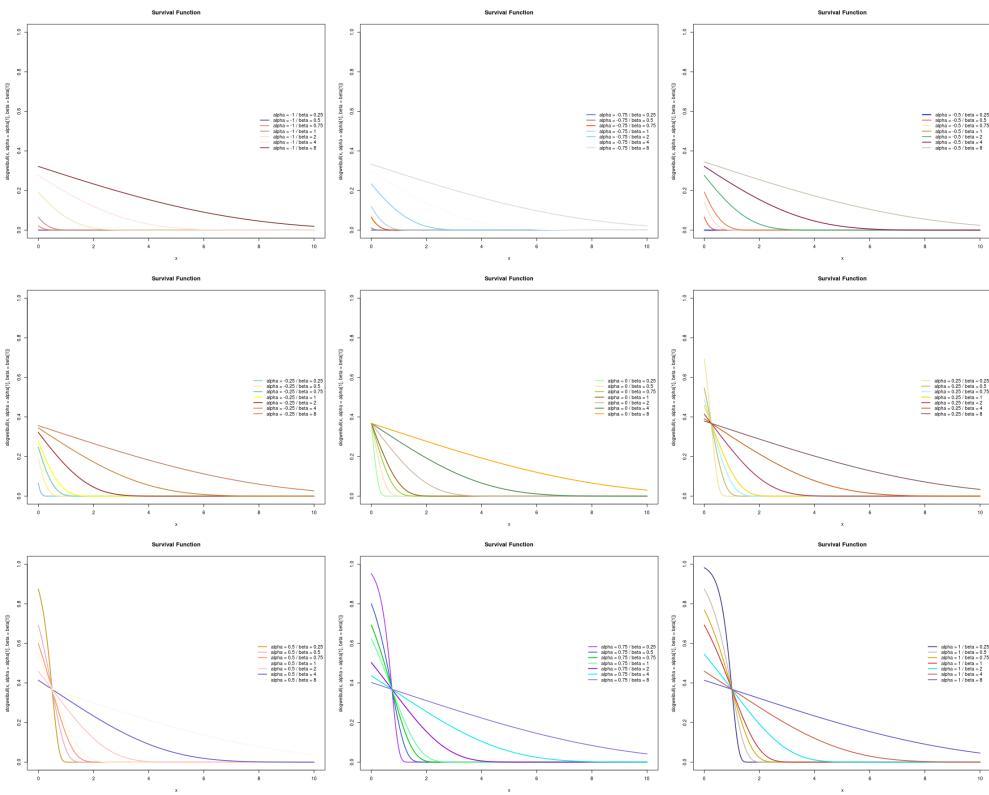


Figure 143: Log-Weibull Distribution에 기반한 생존함수

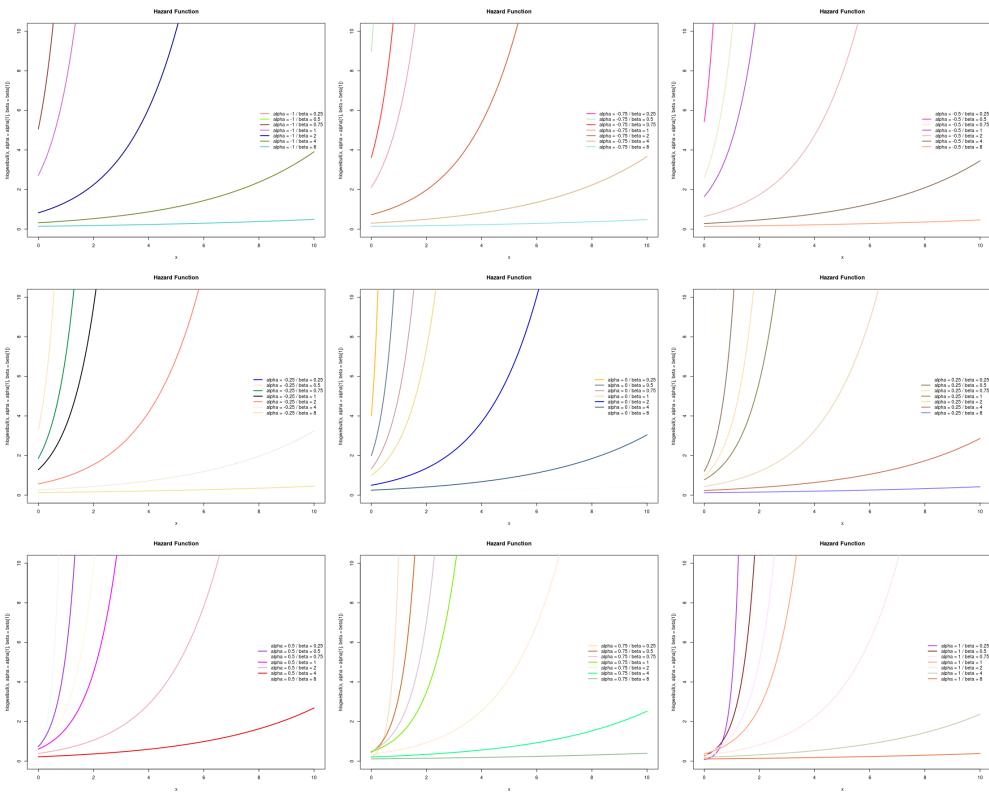


Figure 144: Log-Weibull Distribution에 기반한 위험함수

Code 55. Log-Weibull Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### logweibull Distribution
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9
10 ### input varialbe
11 x = seq(0, 10, length.out = 1000)
12
13
14 ### 수명 분포
15 dlogweibull = function(x, alpha = 0, beta = 1)
16 {
17   temp = (x-alpha)/beta
18   fx = (1/beta) * exp(temp - exp(temp))
19   return(fx)
20 }
21
22
23 ### 누적분포함수
24 plogweibull = function(x, alpha = 0, beta = 1)
25 {
26   fx = -(slogweibull(x, alpha = alpha, beta = beta) - 1)
27   return(fx)
28 }
29
30
31 ### 생존함수
32 slogweibull = function (x, alpha = 0, beta = 1)
33 {
34   temp = (x-alpha)/beta
35   fx = exp(- exp(temp))
36   return(fx)
37 }
38
39
40 ### 위험함수
41 hlogweibull = function (x, alpha = 0, beta = 1)
42 {
43   fx = dlogweibull(x, alpha, beta) / slogweibull(x, alpha, beta)
44   return(fx)
45 }
46
47
48
49
50
51 ##### Plot
52 plot.logweibull_seq = function(x, alpha = 0, beta = 1, xlim=c(0, 10), ylim=c(0, 5), func="dlogweibull")
53 {
54   color=colorPalette(300)
55
56   len_alpha = length(alpha) # alpha 파라메터의 길이
57   len_beta = length(beta) # beta 파라메터의 길이
58
59   color_counter = 1
60   for (i in 1:len_alpha) ### 파라메터: alpha
61   {
62     color_counter_init = color_counter
63     legend_name = NULL;
64
65     if (func=="dlogweibull") # 수명분포
66     {
67       plot(x, dlogweibull(x, alpha=alpha[i], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="Life
68           Distribution Function")
69       for (j in 1:len_beta) ### 파라메터: beta
70       {
71         lines(x, dlogweibull(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
72         color_counter = color_counter + 1;
73         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
74       }
75     } else if (func == "plogweibull") # 누적분포함수
76     {
77       plot(x, plogweibull(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main=
78           "Cumulative Distribution Function")
79       for (j in 1:len_beta) ### 파라메터: beta
80       {
81         lines(x, plogweibull(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
82         color_counter = color_counter + 1;
83         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
84       }
85     } else if (func == "slogweibull") # 생존함수
86     {

```

```

87     plot(x, slogweibull(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
88         Survival Function")
89     for (j in 1:len_beta) ### 파라미터: beta
90     {
91         lines(x, slogweibull(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
92         color_counter = color_counter + 1;
93         legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
94     }
95 } else if (func == "hlogweibull") # 위험함수
96 {
97     plot(x, hlogweibull(x, alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main="
98         Hazard Function")
99     for (j in 1:len_beta) ### 파라미터: beta
100    {
101        lines(x, hlogweibull(x, alpha=alpha[i], beta=beta[j]), col=color[color_counter], lwd=2);
102        color_counter = color_counter + 1;
103        legend_name = c(legend_name, paste("alpha = ", alpha[i], " / beta = ", beta[j], sep=""))
104    }
105 legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
106 }
107 }
108 par(mfrow = c(3, 3))
109 plot.logweibull_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 5), func="dlogweibull")
110
111 par(mfrow = c(3, 3))
112 plot.logweibull_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="plogweibull")
113
114 par(mfrow = c(3, 3))
115 plot.logweibull_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 1), func="slogweibull")
116
117 par(mfrow = c(3, 3))
118 plot.logweibull_seq(x, alpha, beta, xlim=c(min(x), max(x)), ylim=c(0, 10), func="hlogweibull")
119

```

14.39.4 Double Weibull Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{2b} \left \frac{t-a}{b} \right ^{c-1} e^{-\left \frac{t-a}{b} \right ^c}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - \frac{1}{2} e^{-\left(\frac{a-t}{b} \right)^c}$ for $x \leq a$ $\frac{1}{2} e^{-\left(\frac{t-a}{b} \right)^c}$ for $x \geq a$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c \left(\frac{a-t}{b} \right)^{c-1} e^{-\left(\frac{a-t}{b} \right)^c}}{b \left[2 - e^{-\left(\frac{a-t}{b} \right)^c} \right]}$ for $x \leq a$ $\frac{c \left(\frac{t-a}{b} \right)^{c-1} e^{-\left(\frac{t-a}{b} \right)^c}}{b e^{-\left(\frac{t-a}{b} \right)^c}}$ for $x \geq a$	
변수		$t \in \mathbf{R}$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 70: Double Weibull 분포함수에 기반한 척도 함수

만일 $c = 2$ 가 되면, 이 분포는 Laplace 분포가 된다.

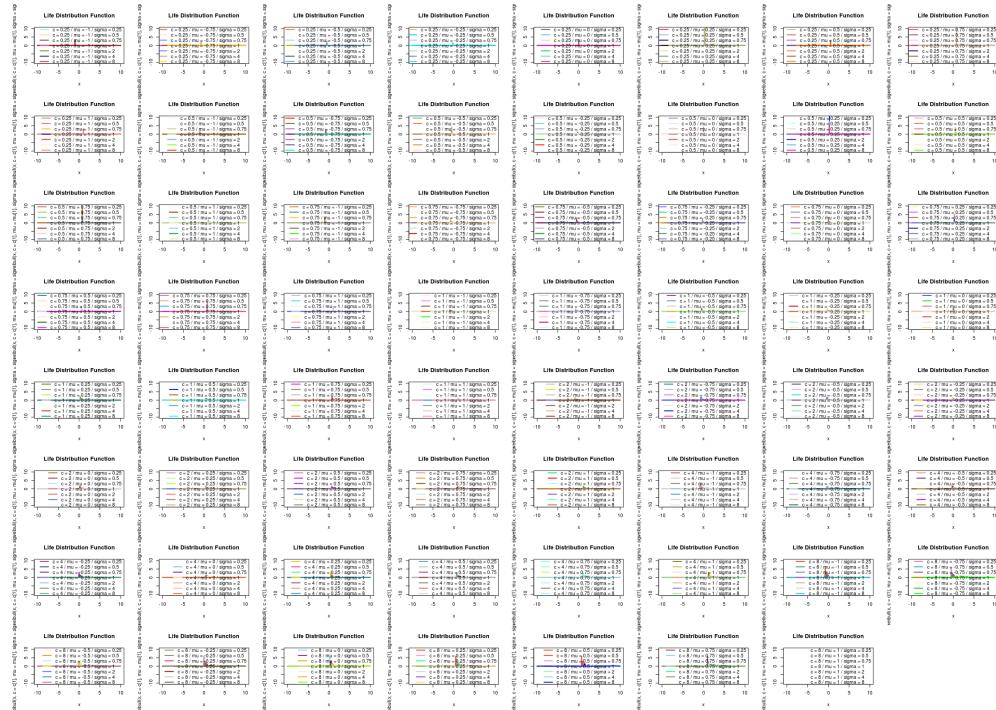


Figure 145: Double Weibull Distribution에 기반한 수명분포함수

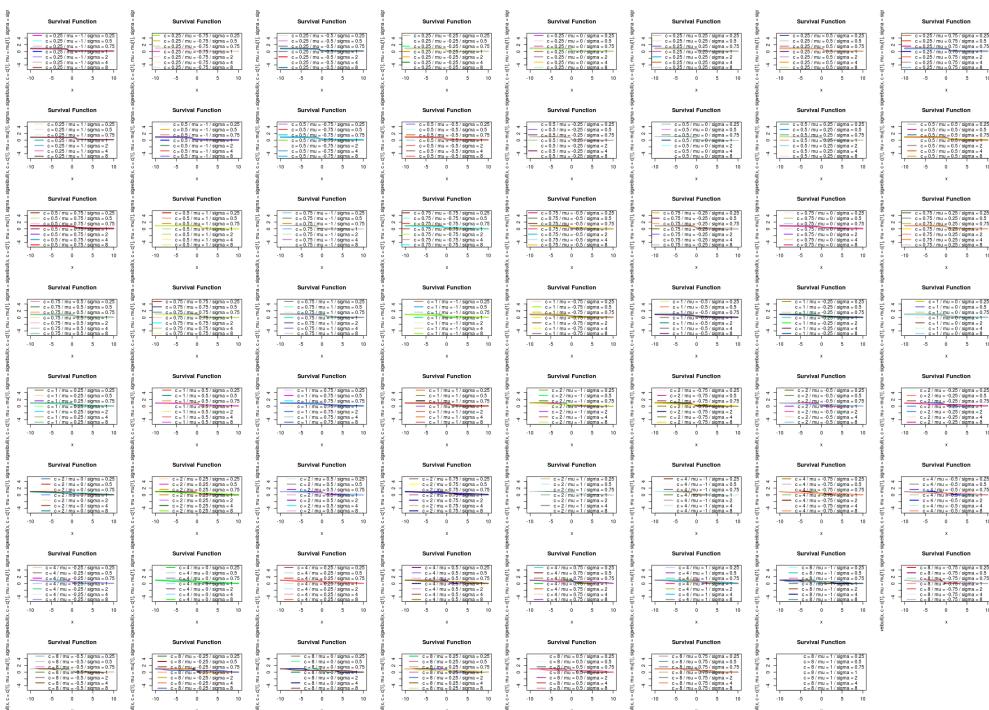


Figure 146: Double Weibull Distribution에 기반한 생존함수



Figure 147: Double Weibull Distribution에 기반한 위험함수

Code 56. Double Weibull Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3 require(VaRES)
4
5 ##### weibull Distribution with c Parameters
6 ### parameter
7 mu = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 sigma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 c = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(-10, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 ddweibull(x, c=1, mu=0, sigma=1)
17
18
19 ### 누적분포함수
20 pdweibull(x, c=1, mu=0, sigma=1)
21
22
23 ### 생존함수
24 sdweibull = function(x, mu=1, sigma=1, c=0)
25 {
26   fx = 1 - pdweibull(x, mu=mu, sigma=sigma, c=c)
27   return(fx)
28 }
29
30
31 ### 위험함수
32 hdweibull = function (x, mu=mu, sigma=sigma, c=0)
33 {
34   fx = ddweibull(x, mu=mu, sigma=sigma, c=c) / sdweibull(x, mu=mu, sigma=sigma, c=c)
35   return(fx)
36 }
37
38
39
40
41
42 ##### Plot
43 plot.dweibull_seq = function(x, mu = 1, sigma = 1, c = 1, xlim=c(0, 10), ylim=c(0, 5), func="ddweibull")
44 {
45   color=colorPalette(300)
46
47   len_c = length(c) # c 파라메터의 길이
48   len_mu = length(mu) # mu 파라메터의 길이
49   len_sigma = length(sigma) # sigma 파라메터의 길이
50
51   color_counter = 1
52   for (i in 1:len_c) ### 파라메터: c
53   {
54     if (func=="ddweibull") # 수명분포
55     {
56       for (j in 1:len_mu) ### 파라메터: mu
57       {
58         color_counter_init = color_counter
59         legend_name = NULL;
60         plot(x, ddweibull(x, c=c[i], mu=mu[j], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
61           ="Life Distribution Function")
62         for (k in 1:len_sigma) ### 파라메터: sigma
63         {
64           lines(x, ddweibull(x, c=c[i], mu=mu[j], sigma=sigma[k]), col=color[color_counter], lwd=2);
65           color_counter = color_counter + 1;
66           legend_name = c(legend_name, paste("c = ", c[i], " / mu = ", mu[j], " / sigma = ", sigma[k], sep=""))
67         }
68         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
69       }
70     }
71     else if (func == "pdweibull") # 누적분포함수
72     {
73       for (j in 1:len_mu) ### 파라메터: mu
74       {
75         color_counter_init = color_counter
76         legend_name = NULL;
77         plot(x, pdweibull(x, c=c[i], mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
78           ="Cumulative Distribution Function")
79         for (k in 1:len_sigma) ### 파라메터: sigma
80         {
81           lines(x, pdweibull(x, c=c[i], mu=mu[j], sigma=sigma[k]), col=color[color_counter], lwd=2);
82           color_counter = color_counter + 1;
83           legend_name = c(legend_name, paste("c = ", c[i], " / mu = ", mu[j], " / sigma = ", sigma[k], sep=""))
84         }
85       }
86     }
87     else if (func == "sdweibull") # 생존함수

```

```

87 {
88     for (j in 1:len_mu) ### 파라미터: mu
89     {
90         color_counter_init = color_counter
91         legend_name = NULL;
92         plot(x, sdweibull(x, c=c[1], mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
93             ="Survival Function")
94         for (k in 1:len_sigma) ### 파라미터: sigma
95         {
96             lines(x, sdweibull(x, c=c[i], mu=mu[j], sigma=sigma[k]), col=color[color_counter], lwd=2);
97             color_counter = color_counter + 1;
98             legend_name = c(legend_name, paste("c = ", c[i], " / mu = ", mu[j], " / sigma = ", sigma[k], sep=""))
99         }
100        legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
101    }
102 }
103 else if (func == "hdweibull") # 위험함수
104 {
105     for (j in 1:len_mu) ### 파라미터: mu
106     {
107         color_counter_init = color_counter
108         legend_name = NULL;
109         plot(x, hdweibull(x, c=c[1], mu=mu[1], sigma=sigma[1]), xlim=xlim, ylim=ylim, col=color[1], lwd=2, type = 'n', main
110             ="Hazard Function")
111         for (k in 1:len_sigma) ### 파라미터: sigma
112         {
113             lines(x, hdweibull(x, c=c[i], mu=mu[j], sigma=sigma[k]), col=color[color_counter], lwd=2);
114             color_counter = color_counter + 1;
115             legend_name = c(legend_name, paste("c = ", c[i], " / mu = ", mu[j], " / sigma = ", sigma[k], sep=""))
116         }
117     }
118 }
119 }
120 par(mfrow = c(9, 8))
121 plot.dweibull_seq(x, mu, sigma, c, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="ddweibull")
122
123 par(mfrow = c(9, 8))
124 plot.dweibull_seq(x, mu, sigma, c, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pdweibull")
125
126 par(mfrow = c(9, 8))
127 plot.dweibull_seq(x, mu, sigma, c, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sdweibull")
128
129 par(mfrow = c(9, 8))
130 plot.dweibull_seq(x, mu, sigma, c, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hdweibull")
131

```

14.39.5 Inverse Weibull Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(\frac{t-a}{b}\right)^{-c-1} e^{-\left(\frac{t-a}{b}\right)^{-c}}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - e^{-\left(\frac{t-a}{b}\right)^{-c}}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c \left(\frac{t-a}{b}\right)^{-c-1} e^{-\left(\frac{t-a}{b}\right)^{-c}}}{b \left[1 - e^{-\left(\frac{t-a}{b}\right)^{-c}}\right]}$	IDHR
변수		$t \geq a$	
파라메터		$a \in \mathbf{R}, b > 0, c > 0$	

Table 71: Inverse Weibull 분포함수에 기반한 척도 함수



Figure 148: Inverse Weibull Distribution에 기반한 수명분포함수

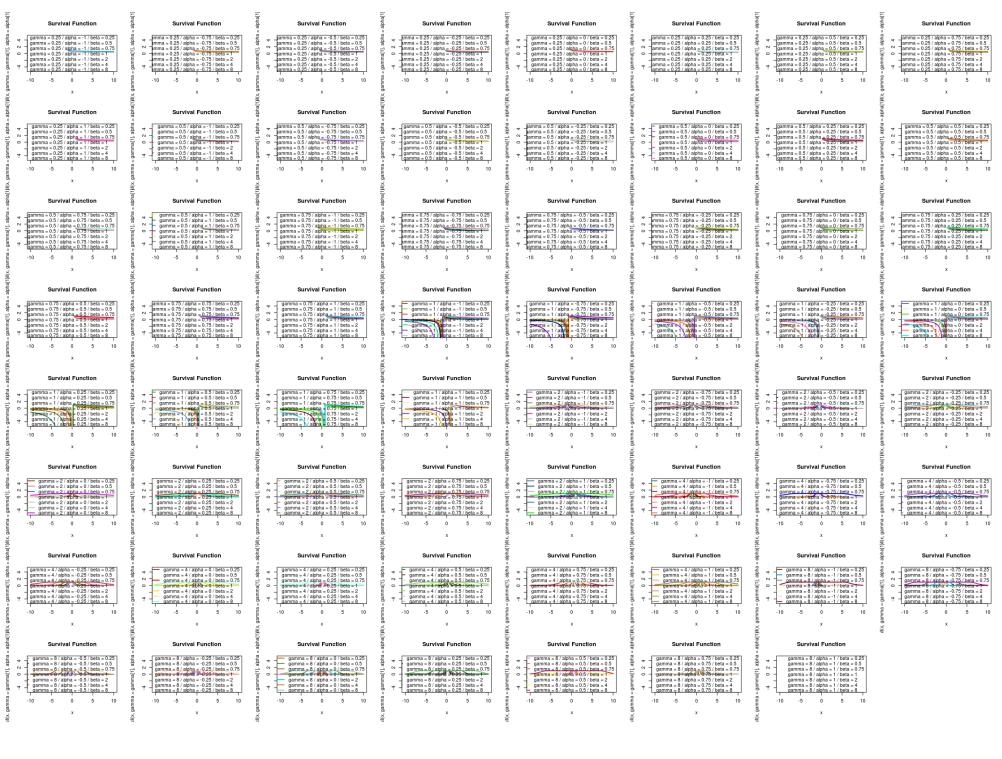


Figure 149: Inverse Weibull Distribution에 기반한 생존함수



Figure 150: Inverse Weibull Distribution에 기반한 위험함수

Code 57. Inverse Weibull Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### weibull Distribution with gamma Parameters
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(-10, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dinverseweibull = function(x, alpha=0, beta=1, gamma=1)
17 {
18   temp = (x-alpha)/beta
19   fx = (gamma/beta) * temp^(-gamma-1) * exp(-temp^(-gamma))
20   return(fx)
21 }
22
23
24 ### 누적분포함수
25 pinverseweibull = function(x, alpha=0, beta=1, gamma=1)
26 {
27   fx = -(sinverseweibull(x, alpha=alpha, beta=beta, gamma=gamma) - 1)
28   return(fx)
29 }
30
31
32 ### 생존함수
33 sinverseweibull = function(x, alpha=0, beta=1, gamma=1)
34 {
35   temp = (x-alpha)/beta
36   fx = 1 - exp(-temp^(-gamma))
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hinverseweibull = function(x, alpha=0, beta=1, gamma=1)
43 {
44   fx = dinverseweibull(x, alpha=alpha, beta=beta, gamma=gamma) / sinverseweibull(x, alpha=alpha, beta=beta, gamma=gamma)
45   return(fx)
46 }
47
48
49
50
51
52 ##### Plot
53 plot.inverseweibull_seq = function(x, alpha = 0, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dinverseweibull")
54 {
55   color=colorPalette(300)
56
57   len_gamma = length(gamma) # gamma 파라미터의 길이
58   len_alpha = length(alpha) # alpha 파라미터의 길이
59   len_beta = length(beta) # beta 파라미터의 길이
60
61   color_counter = 1
62   for (i in 1:len_gamma) ### 파라미터: gamma
63   {
64     if (func=="dinverseweibull") # 수명분포
65     {
66       for (j in 1:len_alpha) ### 파라미터: alpha
67       {
68         color_counter_init = color_counter
69         legend_name = NULL;
70         plot(x, dinverseweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
71           =2, type = 'n', main="Life Distribution Function")
72         for (k in 1:len_beta) ### 파라미터: beta
73         {
74           lines(x, dinverseweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
75           color_counter = color_counter + 1;
76           legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep=""))
77         }
78         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
79       }
80     }
81   else if (func == "pinverseweibull") # 누적분포함수
82   {
83     for (j in 1:len_alpha) ### 파라미터: alpha
84     {
85       color_counter_init = color_counter
86       legend_name = NULL;
87       plot(x, pinverseweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd

```

```

87      =2, type = 'n', main="Cumulative Distribution Function")
88  for (k in 1:len_beta) ### 파라메터: beta
89  {
90    lines(x, pinverseweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
91    color_counter = color_counter + 1;
92    legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep ""))
93  }
94  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
95  }
96 else if (func == "sinverseweibull") # 생존함수
97 {
98  for (j in 1:len_alpha) ### 파라메터: alpha
99  {
100   color_counter_init = color_counter
101   legend_name = NULL;
102   plot(x, sinverseweibull(x, gamma=gamma[1], alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
103   =2, type = 'n', main="Survival Function")
104  for (k in 1:len_beta) ### 파라메터: beta
105  {
106    lines(x, sinverseweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
107    color_counter = color_counter + 1;
108    legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep ""))
109  }
110  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
111  }
112 else if (func == "hinverseweibull") # 위험함수
113 {
114  for (j in 1:len_alpha) ### 파라메터: alpha
115  {
116   color_counter_init = color_counter
117   legend_name = NULL;
118   plot(x, hinverseweibull(x, gamma=gamma[1], alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
119   =2, type = 'n', main="Hazard Function")
120  for (k in 1:len_beta) ### 파라메터: beta
121  {
122    lines(x, hinverseweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
123    color_counter = color_counter + 1;
124    legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep ""))
125  }
126  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
127  }
128  }
129 }
130 par(mfrow = c(9, 8))
131 plot.inverseweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dinverseweibull")
132
133 par(mfrow = c(9, 8))
134 plot.inverseweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="pinverseweibull")
135
136 par(mfrow = c(9, 8))
137 plot.inverseweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sinverseweibull")
138
139 par(mfrow = c(9, 8))
140 plot.inverseweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hinverseweibull")
141

```

14.39.6 Reflected Weibull Distribution

척도 함수	기호	내용	구분
수명분포함수	$f(t)$	$\frac{c}{b} \left(\frac{a-t}{b}\right)^{c-1} e^{-\left(\frac{a-t}{b}\right)^c}$	
생존함수 (신뢰도함수)	$S(t) = P(T > t)$ $= 1 - P(T \leq t)$ $= 1 - F(t)$ $= e^{-H(t)}$	$1 - e^{-\left(\frac{a-t}{b}\right)^c}$	
위험률 함수 (고장률 함수)	$h(t) = \frac{f(t)}{S(t)}$	$\frac{c \left(\frac{a-t}{b}\right)^{c-1}}{b \left[e^{\left(\frac{a-t}{b}\right)^c} - 1\right]}$	IHR
변수 파라메터	$t \leq a$ $a \in \mathbf{R}, b > 0$		

Table 72: Reflected Weibull 분포함수에 기반한 척도 함수



Figure 151: Reflected Weibull Distribution에 기반한 수명분포함수

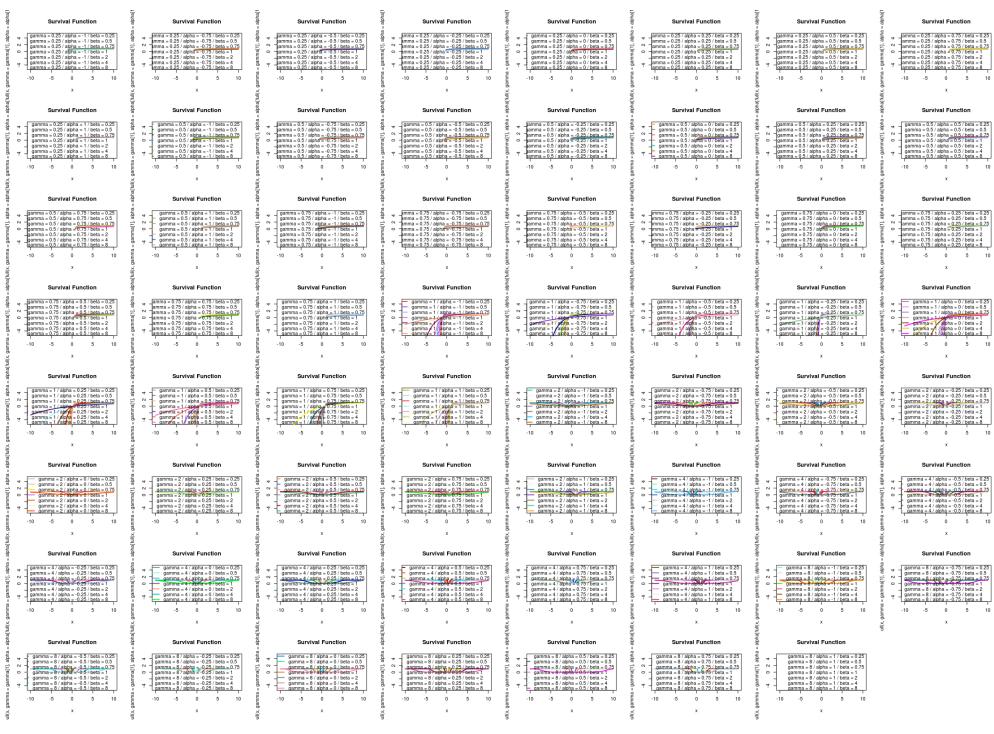


Figure 152: Reflected Weibull Distribution에 기반한 생존함수



Figure 153: Reflected Weibull Distribution에 기반한 위험함수

Code 58. Alpha Distribution에 기반한 수명분포함수, 누적분포함수, 생존함수, 위험률 함수(R 코드)

```

1 setwd("/home/lv999/Dropbox/Github/SurvivalAnalysis/RCode")
2 source("colorPalette.R")
3
4
5 ##### weibull Distribution with gamma Parameters
6 ### parameter
7 alpha = c(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)
8 beta = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
9 gamma = c(0.25, 0.5, 0.75, 1, 2, 4, 8)
10
11 ### input varialbe
12 x = seq(-10, 10, length.out = 1000)
13
14
15 ### 수명 분포
16 dreflectedweibull = function(x, alpha=0, beta=1, gamma=1)
17 {
18   temp = (x-alpha)/beta
19   fx = (gamma/beta) * temp^(gamma-1) * exp(-temp^gamma)
20   return(fx)
21 }
22
23
24 ### 누적분포함수
25 preflectedweibull = function(x, alpha=0, beta=1, gamma=1)
26 {
27   fx = -(sreflectedweibull(x, alpha=alpha, beta=beta, gamma=gamma) - 1)
28   return(fx)
29 }
30
31
32 ### 생존함수
33 sreflectedweibull = function(x, alpha=0, beta=1, gamma=1)
34 {
35   temp = (x-alpha)/beta
36   fx = 1 - exp(-temp^gamma)
37   return(fx)
38 }
39
40
41 ### 위험함수
42 hreflectedweibull = function(x, alpha=0, beta=1, gamma=1)
43 {
44   fx = dreflectedweibull(x, alpha=alpha, beta=beta, gamma=gamma) / sreflectedweibull(x, alpha=alpha, beta=beta, gamma=gamma)
45   return(fx)
46 }
47
48
49
50
51
52 ##### Plot
53 plot.reflectedweibull_seq = function(x, alpha = 0, beta = 1, gamma = 1, xlim=c(0, 10), ylim=c(0, 5), func="dreflectedweibull")
54 {
55   color=colorPalette(300)
56
57   len_gamma = length(gamma) # gamma 파라미터의 길이
58   len_alpha = length(alpha) # alpha 파라미터의 길이
59   len_beta = length(beta) # beta 파라미터의 길이
60
61   color_counter = 1
62   for (i in 1:len_gamma) ### 파라미터: gamma
63   {
64     if (func=="dreflectedweibull") # 수명분포
65     {
66       for (j in 1:len_alpha) ### 파라미터: alpha
67       {
68         color_counter_init = color_counter
69         legend_name = NULL;
70         plot(x, dreflectedweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
71           =2, type = 'n', main="Life Distribution Function")
72         for (k in 1:len_beta) ### 파라미터: beta
73         {
74           lines(x, dreflectedweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
75           color_counter = color_counter + 1;
76           legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep=""))
77         }
78         legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
79       }
80     }
81   else if (func == "preflectedweibull") # 누적분포함수
82   {
83     for (j in 1:len_alpha) ### 파라미터: alpha
84     {
85       color_counter_init = color_counter
86       legend_name = NULL;
87       plot(x, preflectedweibull(x, gamma=gamma[1], alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd

```

```

87      =2, type = 'n', main="Cumulative Distribution Function")
88  for (k in 1:len_beta) ### 파라미터: beta
89  {
90    lines(x, preflectedweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
91    color_counter = color_counter + 1;
92    legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep=""))
93  }
94  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
95  }
96 else if (func == "sreflectedweibull") # 생존함수
97 {
98  for (j in 1:len_alpha) ### 파라미터: alpha
99  {
100   color_counter_init = color_counter
101   legend_name = NULL;
102   plot(x, sreflectedweibull(x, gamma=gamma[1], alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
103   =2, type = 'n', main="Survival Function")
104   for (k in 1:len_beta) ### 파라미터: beta
105   {
106     lines(x, sreflectedweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
107     color_counter = color_counter + 1;
108     legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep=""))
109   }
110  legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
111  }
112 else if (func == "hreflectedweibull") # 위험함수
113 {
114  for (j in 1:len_alpha) ### 파라미터: alpha
115  {
116   color_counter_init = color_counter
117   legend_name = NULL;
118   plot(x, hreflectedweibull(x, gamma=gamma[1], alpha=alpha[1], beta=beta[1]), xlim=xlim, ylim=ylim, col=color[1], lwd
119   =2, type = 'n', main="Hazard Function")
120   for (k in 1:len_beta) ### 파라미터: beta
121   {
122     lines(x, hreflectedweibull(x, gamma=gamma[i], alpha=alpha[j], beta=beta[k]), col=color[color_counter], lwd=2);
123     color_counter = color_counter + 1;
124     legend_name = c(legend_name, paste("gamma = ", gamma[i], " / alpha = ", alpha[j], " / beta = ", beta[k], sep=""))
125   }
126   legend('right', bty = 'n', lwd=2, col=color[color_counter_init:(color_counter - 1)], legend = legend_name)
127  }
128  }
129 }
130 par(mfrow = c(9, 8))
131 plot.reflectedweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-10, 10), func="dreflectedweibull")
132
133 par(mfrow = c(9, 8))
134 plot.reflectedweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="preflectedweibull")
135
136 par(mfrow = c(9, 8))
137 plot.reflectedweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-5, 5), func="sreflectedweibull")
138
139 par(mfrow = c(9, 8))
140 plot.reflectedweibull_seq(x, alpha, beta, gamma, xlim=c(min(x), max(x)), ylim=c(-30, 30), func="hreflectedweibull")
141

```

14.40 Wigner's Semi-circle Distribution

이 분포는 Semi-elliptical distribution으로도 알려져 있다. Chapter 14.32를 참고하도록 하자.

Part VII

Cox 비례위험모형

만일 생존시간 데이터의 분포에 대한 정보가 있다면, 모수회귀모형을 이용하면 된다.

그러나 이러한 분포에 대한 가정이 적절하지 못할 경우에, 모수회귀모형을 사용하면, 추정된 계수에 대한 정확성에 대해 신뢰성을 잃게 된다.

모수회귀모형과 비례위험모형의 차이점을 간략하게 보면 다음과 같다.

모수회귀모형:

$$\log T_i = Z_i \beta + \epsilon_i$$

Covariate Z_i 가 반응변수 $\log T_i$ 와 선형관계로 표현된다.

비례위험모형:

$$h(t|Z_i) = h_0(t) \exp(Z_i \beta) \quad (16)$$

* 회귀계수 β 가 covariate 변화에 따른 log(생존시간)의 관계를 나타낸다는 점에 있어서는 동일하다.

비례위험모형(proportional hazard model, 이하 PH model)은, 위험함수에 covariate에 대한 회귀식을 포함하는 모형으로, 다음과 같은 특징이 있어 널리 사용되고 있다.

- 기저 분포에 대한 가정이 필요하지 않다.
- time-varying covariate을 회귀모형에 포함시킬 수 있다.
- 추정된 회귀계수는 covariate과 위험함수의 관계를 나타낸다.

Cox 모형의 기본 가정은 다음과 같다.

- Relative hazard는 시간에 관계 없이, 시간과는 독립적으로 일정하다는 비례 위험의 가정

식 (16)으로부터 $\frac{h_x(t)}{h_0(t)} = e^{Z_i \beta}$ 가 되는데, $e^{Z_i \beta}$ 가 상수값처럼 취급된다고 했기 때문이다.

이를 확인하기 위해, Kaplan-Meier 생존곡선을 그려서, 곡선이 교차하지 않고 평행하게 가는지를 확인하여야 한다. 여기에서 relative hazard가 상수라고 해도, 반드시 기차길처럼 평행할 필요는 없다. 생존곡선에서 두 개의 곡선의 차이가 점차적으로 증가하면 비례가정을 만족하는 것이고, 두 개의 곡선이 서로 교차하거나 만나면 비례가정에 위배되는 것이다. 이에 대한 판단은 생존 곡선을 보고 주관적으로 할 수밖에 없다.

이외에도 log minus survival plot을 그려보고, 두 곡선 사이에 일정하게 수직으로 차이가 있으면, 비례 가정을 만족한다고 판단할 수도 있다.

이 비례가정이 만족되지 않는 경우, Cox 모형을 이용할 수 없으며, 비례가정을 만족하지 않을 경우 time dependent covariate approach²⁶를 사용하여야 한다.

- Hazard function과 covariate 사이의 log-linear 관계

hazard function과 covariate 사이에는 기본적으로 log-linear 관계가 있어야 한다. 즉, hazard function에 자연로그를 취한 값은, 각 covariate의 회귀계수와는 linear relation(선형 관계)가 있어야 한다.

이를 확인하기 위해서는 cumulative hazard function으로 Martingale residuals를 구하여 확인할 수 있다. Martingale residual과 covariate가 웬만큼 직선으로 그려지면, 가정을 만족한다고 판단한다.

²⁶가령 Extended Cox 모형 등을 이용할 수 있다.

15 Covariate가 1개인 비례위험모형

15.1 데이터 구조 - covariate가 1개인 경우

Covariate 1개를 포함하므로, 데이터는 $(T_i, \delta_i, Z_i(t))$ ($i = 1, \dots, n$)으로 표기한다.
여기서

- T_i 는 i 번째 object(개인)가 연구에 참여해있는 시간
- δ_i 는 i 번째 object에 대한 사건발생/중도절단 지시변수
$$\delta_i = \begin{cases} 1 & \text{관측된 경우} \\ 0 & \text{중도절단된 경우} \end{cases}$$
- $Z_i(t)$ 는 t 시점에서 i 번째 object에 대한 위험인자 또는 covariate이다.
- $t_1 < \dots < t_n$ 은 순서대로 정렬한 순서통계량

15.2 준모수적 Cox 비례위험모형

Cox 비례위험모형은 covariate에서는 모수적 형태를 가정하고, 기저함수에는 모수적 가정을 가정하지 않으므로 준모수적(semi-parametric) 방법으로 불린다.

t 시점에서 위험인자 Z 를 가진 i 번째 object에 대한 위험함수는 다음과 같다.

$$\begin{aligned} h(t|Z_i) &= h_0(t)\psi(Z; \beta) \\ &= h_0(t)\exp(Z; \beta) \end{aligned}$$

- $h(t|Z)$ 는 t 시점에서 위험인자 또는 covariate Z 를 가진 object에 대한 위험률이다.
- $h_0(t)$ 는 분포 가정이 주어져 있지 않은 기저위험함수(baseline hazard function)이다.
일반적으로 $\psi(Z; \beta) = \exp(Z; \beta)$ 인 지수함수를 고려한다.
- Covariate Z_i 가 1단위 증가할 때마다 $\exp(\beta)$ 만큼 위험률이 증가한다.
위험률은 음수가 될 수 없으므로, 지수함수(음수를 갖지 않는)가 음이 아닌 함수값을 보장해주는 중요한 역할을 하며 널리 이용된다.
 - 그 외에는
선형함수 $\psi(Z) = 1 + Z\beta$,
로지스틱 함수 $\psi(Z) = \log(1 + e^{Z\beta})$
등이 있다. 자세한 것은, Chapter 14를 참고하도록 하자.

1개의 covariate를 고려한 경우, 두 object 간 사건 발생 위험비를 비교해 보자.
object i 와 object j 의 사건발생 위험비는

$$\begin{aligned} \frac{h(t|Z_i)}{h(t|Z_j)} &= \frac{h_0(t)\exp(Z_i\beta)}{h_0(t)\exp(Z_j\beta)} \\ &= \exp[(Z_i - Z_j)\beta] \end{aligned}$$

즉, 이러한 위험비는 시간에 의존하지 않고, 회귀계수 β 와 covariate 값의 차이 $(Z_i - Z_j)$ 에 의존한다는 것을 알 수 있다.

Covariate이 $Z = 1$ 또는 $Z = 0$ 인 binary 데이터고 하자.

두 object i 와 j 에 대해 covariate $Z_i = 1, Z_j = 0$ 인 경우, 사건 발생 위험비는

$$\begin{aligned} \frac{h(t|Z_i)}{h(t|Z_j)} &= \frac{h_0(t)\exp(Z_i\beta)}{h_0(t)\exp(Z_j\beta)} \\ &= \exp[(Z_i - Z_j)\beta] \\ &= \exp(\beta) \end{aligned}$$

이러한 위험비는 $\exp(\beta)$ 는 시간이 변하더라도 불변(invariant)으로, 시점에 의존하지 않는다.
그래서 Cox가 ”비례위험(proportional hazard)”이라고 불렀다.

15.3 회귀계수에 대한 추론

회귀계수 β 를 추정하기 위해서는 부분가능도함수(partial likelihood function)를 이용한다.

가능도함수는 관측 데이터셋의 확률에 비례하지만, 반면 부분가능도함수는 그렇지 않다. 그럼에도 불구하고, 부분가능도함수는 근사적인 추론에서 가능도함수와 마찬가지로 다를 수 있어서 유용하다.



동점이 없는 생존시간 데이터에 대한 부분가능도함수는

$$\begin{aligned} PL(\beta) &= \prod_{i=1}^n \left[\frac{Y_i \exp(Z_i \beta)}{\sum_{\ell \in R_i} Y_\ell \exp(Z_\ell \beta)} \right]^{\delta_i} \\ &= \prod_{i=1}^n \left[\frac{Y_i r_i(\beta, t)}{\sum_{\ell \in R_i} Y_\ell r_\ell(\beta, t)} \right]^{\delta_i} \end{aligned}$$

여기서, $r_i(\beta, t) = \exp[Z_i \beta] = r_i(t)$ 은 i 번째 object에 대한 위험점수(risk score)이다.



여기서 로그를 취하면

$$\begin{aligned} p\ell(\beta) &= \log PL(\beta) \\ &= \sum_{i=1}^n \delta_i \left[\log \{Y_i \exp(Z_i \beta)\} - \log \left\{ \sum_{\ell \in R_i} Y_\ell \exp(Z_\ell \beta) \right\} \right] \end{aligned}$$



로그 부분가능도함수 $p\ell(\beta)$ 를 β 에 대해 미분하면, score function $U(\beta)$ 를 얻을 수 있다.

$$\begin{aligned} U(\beta) &= \frac{\partial [p\ell(\beta)]}{\partial \beta} \\ &= \sum_{i=1}^n \delta_i \left[\frac{Z_i Y_i \exp(Z_i \beta)}{Y_i \exp(Z_i \beta)} - \frac{\sum_{\ell \in R_i} Z_\ell Y_\ell \exp(Z_\ell \beta)}{\sum_{\ell \in R_i} Y_\ell \exp(Z_\ell \beta)} \right] \\ &= \sum_{i=1}^n \delta_i \left[Z_i - \frac{\sum_{\ell \in R_i} Z_\ell Y_\ell \exp(Z_\ell \beta)}{\sum_{\ell \in R_i} Y_\ell \exp(Z_\ell \beta)} \right] \end{aligned}$$

최대부분가능도추정량 $\hat{\beta}$ 은 다음의 부분가능도방정식을 풀어 구한다.

$$U(\hat{\beta}) = 0$$



이렇게 구한 해 $\hat{\beta}$ 은 β 에 대한 일치추정량이 되며, 근사적으로 정규분포 $N(\beta, E[I(\beta)]^{-1})$ 를 따른다. 여기서 information matrix $I(\beta)$ 는

$$\begin{aligned} I(\beta) &= -\frac{\partial^2 [p\ell(\beta)]}{\partial \beta^2} \\ &= \sum_{i=1}^n \delta_i \left[\frac{Z_i Y_i \exp(Z_i \beta)}{Y_i \exp(Z_i \beta)} - \frac{\sum_{\ell \in R_i} Z_\ell Y_\ell \exp(Z_\ell \beta)}{\sum_{\ell \in R_i} Y_\ell \exp(Z_\ell \beta)} \right] \end{aligned}$$



함수 $U(\beta) = \frac{\partial \ell(\beta)}{\partial \beta}$ 에 대해 테일러 공식을 적용하면, 근사적으로

$$\begin{aligned} 0 &= U(\hat{\beta}) \\ &\approx U(\beta) + U'(\beta)(\hat{\beta} - \beta) \end{aligned}$$



$U'(\beta) = \frac{\partial^2 [p\ell(\beta)]}{\partial \beta^2}$ 이고, $-U'(\beta) = I(\beta)$ 으로,

$$\hat{\beta} = \beta[U'(\beta)]^{-1}$$

$$U(\beta) \approx \beta + I^{-1}(\beta) \cdot U(\beta)$$



적당한 초기값 $\hat{\beta}^{(0)}$ 를 이용하여, 다음과 같이 Newton-Raphson 알고리즘을 사용하여 반복적인 계산을 통해 $\hat{\beta}$ 를 구한다.

$$\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} + I^{-1}(\hat{\beta}^{(k)})U(\hat{\beta}^{(k)})$$

로그부분가능도함수가 수렴할 때까지, 즉 $\ell(\hat{\beta}^{(k+1)}) \approx \ell(\hat{\beta}^{(k)})$ 일 때까지 반복적인 계산을 하여 해를 구한다.

✓ 이렇게 구한 최대부분가능도추정량 $\hat{\beta}$ 을 기반으로 누적위험함수를 구하기 위해, 다음과 같은 Breslow 추정량을 이용한다.

$$\hat{H}(t|Z) = \sum_{t_i \leq t} \frac{d_i}{\sum_{\ell \in R_i} \exp(Z_\ell \hat{\beta})}$$

여기서, $d_i = \sum_{\ell=1}^n I(t_\ell = t_i)$

✓ 회귀계수 추정량 $\hat{\beta}$ 의 분포는, 마팅게일 중심극한정리를 이용한다.

$$\hat{\beta} - \beta \sim N(0, I^{-1}(\beta))$$

회귀계수추정량 $\hat{\beta}$ 은 근사적으로 정규분포를 따른다.

✓ 회귀계수 β 에 대한 근사적인 $(1 - \alpha) \times 100\%$ 신뢰구간은 다음과 같다.

$$\hat{\beta} \pm z_{\alpha/2} [I^{-1}(\beta)]^{\frac{1}{2}}$$

16 Covariate가 여러 개인 비례위험모형

16.1 데이터 구조 - covariate가 여러 개인 경우

Covariate vector를 포함하므로, 데이터는 $(T_i, \delta_i, Z_i(t))$ (단, $i = 1, \dots, n$)으로 표기한다.
여기서

- T_i 는 i 번째 object(개인)가 연구에 참여해있는 시간
- δ_i 는 i 번째 object에 대한 사건발생/중도절단 지시변수
$$\delta_i = \begin{cases} 1 & \text{관측된 경우} \\ 0 & \text{중도절단된 경우} \end{cases}$$
- $Z_i(t) = [Z_{i1}(t), \dots, Z_{ip}(t)]'$ 는 t 시점에서 i 번째 object에 대한 위험인자 또는 covariate
 $Z_{ij}(t)$ 는 t 시점에서 i 번째 object의 j 번째 위험인자 또는 covariate
Covariate가 시간 t 에 의존하지 않을 경우에는 $Z_{ij}(t) = Z_{ij}$ 로 표기하기도 한다.

16.2 준모수적 Cox 비례위험모형

Cox 비례위험모형은 covariate에서는 모수적 형태를 가정하고, 기저함수에는 모수적 가정을 가정하지 않으므로 준모수적(semi-parametric) 방법으로 불린다.

t 시점에서 위험인자벡터 $Z = [Z_{i1}(t), \dots, Z_{ip}(t)]'$ 를 가진 i 번째 object에 대한 위험함수는 다음과 같다.

$$\begin{aligned} h(t|Z_i) &= h_0(t) \exp(Z_i' \beta) \\ &= h_0(t) \exp(Z_{i1}\beta_1 + \dots + Z_{ip}\beta_p) \end{aligned}$$

- $h(t|Z)$ 는 t 시점에서 위험인자 또는 covariate Z 를 가진 object에 대한 위험률이다.
- $h_0(t)$ 는 분포 가정이 주어져 있지 않은 기저위험함수(baseline hazard function)이다.
- $\beta = (\beta_1, \dots, \beta_p)'$ 는 $p \times 1$ 회귀계수벡터로, covariate의 효과를 추정하는 회귀계수로 구성된다.
- Covariate Z_i 가 1단위 증가할 때마다 $\exp(\beta)$ 만큼 위험률이 증가한다.

p개의 covariate를 고려한 경우, 두 object 간 사건 발생 위험비를 비교해 보자.
object i 와 object j 의 사건발생 위험비는

$$\begin{aligned} \frac{h(t|Z_i)}{h(t|Z_j)} &= \frac{h_0(t) \exp(Z_{i1}\beta_1 + \dots + Z_{ip}\beta_p)}{h_0(t) \exp(Z_{j1}\beta_1 + \dots + Z_{jp}\beta_p)} \\ &= \exp \left[\sum_{k=1}^p (Z_{ik} - Z_{jk})\beta_k \right] \end{aligned}$$

즉, 이러한 위험비는 시간에 의존하지 않고, covariate 값에 비례한다.

또한, 이 식을, 위험인자 $Z_i = (Z_{i1}, \dots, Z_{ip})'$ 와 $Z_j = (Z_{j1}, \dots, Z_{jp})'$ 를 가진 object 간의 상대 위험도(relative risk) 또는 위험비(hazard ratio)라고 부른다.

16.3 회귀계수와 누적위험함수의 추정

Covariate가 여러 개인 경우, covariate vector를 이용해 데이터 구조와 모형을 $(T_i, \delta_i, Z_i(t))$, $Z_i = (Z_{i1}, \dots, Z_{ip})'$ (단, $i = 1, \dots, n$)으로 표기한다.

- $\delta_i = \begin{cases} 1 & \text{관측된 경우} \\ 0 & \text{중도절단된 경우} \end{cases}$
- $Z_i = (Z_{i1}, \dots, Z_{ip})'$ 는 $p \times 1$ covariate vector
- $t_1 < \dots < t_n$ 은 순서대로 정렬한 순서통계량
- t_j 시점에 사건을 가진 object의 covariate는 Z_j 로 표기한다.

위험그룹 $R_j = R(t_j)$ 는 t_j 바로 전 시점까지 사건을 경험하지 않고 살아있는 object들의 그룹이다.

16.3.1 비례위험모형

비례위험모형에서 Cox의 비례위험함수는 다음과 같다.

$$\begin{aligned} h(t|Z_i) &= h_0(t) \exp(Z\beta) \\ &= h_0(t) \exp(Z_1\beta_1 + \cdots + Z_p\beta_p) \end{aligned} \quad (17)$$

- $h(t|Z)$ 는 t 시점에서 위험인자 또는 covariate Z 를 가진 object에 대한 위험률이다.
- $h_0(t)$ 는 분포 가정이 주어져 있지 않은 기저위험함수(baseline hazard function)이다.
- $\beta = (\beta_1, \dots, \beta_p)'$ 는 $p \times 1$ 회귀계수벡터로, covariate의 효과를 추정하는 회귀계수로 구성된다.
- $Z = (Z_i, \dots, Z_p)'$ 는 $p \times 1$ covariate vector

i 번째 object에 대한 위험함수는 다음과 같다.

$$\begin{aligned} h(t|Z_i) &= h_0(t) \exp(Z'_i\beta) \\ &= h_0(t) \exp(Z_{i1}\beta_1 + \cdots + Z_{ip}\beta_p) \end{aligned}$$

16.3.2 부분가능도함수

한 object에 대한 부분가능도함수(partial likelihood)는 다음과 같이 구한다.

$$\begin{aligned} L_j^*(\beta) &= P(A|B) \\ &= \frac{h_0(t_j) \exp(Z_j\beta)}{\sum_{\ell \in R(t_j)} h_0(t_\ell) \exp(Z_\ell\beta)} \\ &= \frac{\exp(Z_j\beta)}{\sum_{\ell \in R(t_j)} \exp(Z_\ell\beta)} \end{aligned}$$

✓ 생존시간 데이터에 대한 부분가능도함수는 다음과 같다.

$$\begin{aligned} PL(\beta) &= \prod_{i=1}^n \left[\frac{Y_i \exp(Z'_i\beta)}{\sum_{\ell \in R_i} Y_\ell \exp(Z'_\ell\beta)} \right]^{\delta_i} \\ &= \prod_{i=1}^n \left[\frac{Y_i r_i(\beta, t)}{\sum_{\ell \in R_i} Y_\ell r_\ell(\beta, t)} \right]^{\delta_i} \\ &= \prod_{i=1}^n \left[\frac{Y_i (Z_{i1}\beta_1 + \cdots + Z_{ip}\beta_p)}{\sum_{\ell \in R_i} Y_\ell (Z_{\ell 1}\beta_1 + \cdots + Z_{\ell p}\beta_p)} \right]^{\delta_i} \\ &= \prod_{i=1}^n [L_i^*]^{\delta_i} \end{aligned} \quad (18)$$

여기서 $r_i(\beta, t) = \exp[Z'_i\beta] \equiv r_i(t)$ (i 번째 object에 대한 위험접수)

✓ $p\ell(\beta) = \log PL(\beta)$ 을 이용하여 score function $U(\beta_k)$ 를 다음과 같이 구한다.

$$\begin{aligned} U(\beta_k) &= \frac{\partial[p\ell(\beta)]}{\partial \beta_k} \\ &= \sum_{i=1}^n \delta_i \left[Y_i Z_{ik} - \frac{\sum_{\ell \in R_i} Y_\ell \exp(Z'_\ell\beta) Z_{\ell k}}{\sum_{\ell \in R_i} Y_\ell \exp(Z'_\ell\beta)} \right] \end{aligned}$$

16.3.3 정보행렬(information matrix)

모수에 대한 -2차 미분을 구하여 다음과 같은 정보행렬 $I(\beta)$ 을 얻는다.

$$I(\beta) = [I_{gh}(\beta)]_{p \times p}$$

$$= - \begin{bmatrix} \frac{\partial^2 \ell(\beta)}{\partial \beta_1^2} & \frac{\partial^2 \ell(\beta)}{\partial \beta_1 \partial \beta_2} & \cdots & \frac{\partial^2 \ell(\beta)}{\partial \beta_1 \partial \beta_p} \\ \cdots & \frac{\partial^2 \ell(\beta)}{\partial \beta_2^2} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \frac{\partial^2 \ell(\beta)}{\partial \beta_p^2} \end{bmatrix}$$

여기서 $g, h = 1, \dots, p$ 에 대하여,

$$\begin{aligned} I_{gh}(\beta) &= \frac{\partial^2 \ell(\beta)}{\partial \beta_g \partial \beta_h} \\ &= \sum_{i=1}^n \frac{\sum_{j \in R(t_i)} Z_{jg} z_{jh} \exp[\sum_{k=1}^p \beta_k Z_{jk}]}{\sum_{j \in R(t_i)} \exp[\sum_{k=1}^p \beta_k Z_{jk}]} \\ &\quad - \sum_{i=1}^D \left[\frac{\sum_{j \in R(t_i)} z_{jh} \exp(\sum_{k=1}^p \beta_k z_{jk})}{\sum_{j \in R(t_i)} \exp(\sum_{k=1}^p \beta_k z_{jk})} \right] \left[\frac{\sum_{j \in R(t_i)} z_{jh} \exp(\sum_{k=1}^p \beta_k z_{jk})}{\sum_{j \in R(t_i)} \exp(\sum_{k=1}^p \beta_k z_{jk})} \right] \end{aligned}$$

✓ 적당한 초기값 $\hat{\beta}^{(0)}$ 를 이용하여, 다음과 같이 Newton-Raphson 알고리즘을 사용하여, 반복적인 계산을 통해 $\hat{\beta}$ 를 구한다.

$$\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} + I^{-1}(\hat{\beta}^{(k)})U(\hat{\beta}^{(k)})$$

로그부분가능도함수가 수렴할 때까지, 즉 $\ell(\hat{\beta}^{(k+1)}) \approx \ell(\hat{\beta}^{(k)})$ 일 때까지 반복적인 계산을 하여 해를 구한다.

✓ 이렇게 구한 최대부분가능도추정량 $\hat{\beta}$ 을 기반으로 누적위험함수를 구하기 위해, 다음과 같은 Breslow 추정량을 이용한다.

$$\hat{H}(t|Z) = \sum_{t_i \leq t} \frac{d_i}{\sum_{\ell \in R_i} \exp(Z_\ell \hat{\beta})}$$

여기서, $d_i = \sum_{\ell=1}^n \delta_i I(t_\ell = t_i)$

Note 27. Cox 비례위험모형 추정량의 통계적 성질

이러한 회귀계수추정량 $\hat{\beta}$ 는 다음의 통계적 성질을 만족한다.

[일치성(consistency)] 표본의 크기가 클 수록 $\hat{\beta}$ 은 참값 β 로 수렴한다.

[근사적 정규성(asymptotic normality)] $\hat{\beta}$ 는 근사적으로 정규분포를 따른다.

[효율성(efficiency)] β 에 대한 추정량들 중 MPLE(maximum partial likelihood estimator)가 최소분산을 갖는다.

17 동점 처리(handling ties)

비례위험모형에서 유도된 부분가능도함수 식은, 모든 시점들이 서로 동일하지 않다는 가정 하에서 유도된다.

그러나 실제 데이터에서 생존시간이 동일한 경우는 종종 발생할 것이다.

이러한 동점 생존시간이 존재할 경우, 부분가능도함수를 생성하는 여러가지 동점 처리 방법들이 제안되었다.

설명의 편의를 위해, 예를 하나 만들어놓자. 다음의 데이터셋에서 부분가능도함수를 계산해보자. 환자 1과 환자 2의 시간 데이터 값이 동점인 경우이다.

($t_1 = t_2 < t_3 < t_4 < t_5$ 라고 하자.)

ID	t	δ	Z
1	t_1	1	z_1
2	t_2	1	z_2
3	t_3	0	z_3
4	t_4	1	z_4
5	t_5	1	z_5

환자 1과 환자 2 사이의 순서를 알 수 없으므로, 가능한 순서는 $2!$ 이다.

- A_1 은 환자 1이 환자 2보다 먼저 사건이 발생하는 경우
- A_2 은 환자 2이 환자 1보다 먼저 사건이 발생하는 경우

그러면, $L_1(\beta) = P(A_1 \cup A_2) = P(A_1) + P(A_2)$ 가 된다.

17.1 Exact Method

가능한 위험집합에서 부분가능도함수를 계산하여 동점에 대해 처리하는 방법을 말한다.

이제 앞서 만든 예를 바탕으로 exact method를 생각해보자. 동점인 두 관측값 $t_1 = t_2$ 의 부분가능도함수 기여값은 다음과 같이 계산할 수 있다.

$$P(A_1) = \left(\frac{e^{z_1\beta}}{e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right) \left(\frac{e^{z_2\beta}}{e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right)$$

$$P(A_2) = \left(\frac{e^{z_2\beta}}{e^{z_2\beta} + e^{z_1\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right) \left(\frac{e^{z_1\beta}}{e^{z_1\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right)$$

$L_j(\beta)$ 를 j 번째 서로 다른 값 $\{t_1, t_4, t_5\}$ 에서의 부분가능도함수라고 할 때 (t_3 는 중도절단이므로, 부분가능도계산에서 제외됨)

$$L(\beta) = L_1(\beta)L_2(\beta)L_3(\beta)$$

여기서 $L_1(\beta) = P(A_1) + P(A_2)$ 로 계산한다.

일단 부분가능도함수가 계산된 후에는 모두 추정 방법은 동일하다.

$$L_2(\beta) = \frac{e^{z_1\beta}}{e^{z_1\beta} + e^{z_5\beta}}$$

$$L_3(\beta) = \frac{e^{z_5\beta}}{e^{z_5\beta}} = 1$$

17.2 Breslow's Approximation

Breslow가 제안한 통계량으로, 근사 정도가 좋은 경우에 유용하며, 매우 간단한 방법이다.

Exact method에서 사용한 식(1)을 식을 다음과 같은 근사를 이용하여 $P(A_1)$, $P(A_2)$ 를 계산한다.

$$\left(\frac{e^{z_2\beta}}{e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right) \approx \left(\frac{e^{z_2\beta}}{e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right)$$

$$\left(\frac{e^{z_1\beta}}{e^{z_1\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right) \approx \left(\frac{e^{z_1\beta}}{e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right)$$

⇒

$$P(A_1) = \left(\frac{e^{z_1\beta}}{e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right) \left(\frac{e^{z_2\beta}}{e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right)$$

$$= \frac{e^{(z_1+z_2)\beta}}{[e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}]^2}$$

$$P(A_2) = \left(\frac{e^{z_2\beta}}{e^{z_2\beta} + e^{z_1\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right) \left(\frac{e^{z_1\beta}}{e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}} \right)$$

$$= \frac{e^{(z_1+z_2)\beta}}{[e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta}]^2}$$

따라서,

$$L_1(\beta) \approx \frac{\exp \left[\beta \sum_{\ell \in R_i} z_\ell \right]}{\left[\sum_{\ell \in R_i} \exp(z_\ell \beta) \right]^{d_j}}$$

즉, 동점값이 모두 분모에 들어가게 되어

$$\left(\frac{r_1}{r_1 + r_2 + r_3 + r_4 + r_5} \right) \left(\frac{r_2}{r_1 + r_2 + r_3 + r_4 + r_5} \right)$$

으로 나타낸다.

17.3 Efron's Approximation

부분가능도에 가까운 근사법이다.

Exact method에 의하면 다음과 같이 쓸 수 있다.

$$\begin{aligned} L_1(\beta) &= \frac{bc}{a(a-b)} + \frac{bc}{a(a-c)} \\ &\approx \frac{2bc}{a} \frac{1}{\left(a - \frac{b+c}{2}\right)} \end{aligned}$$

여기서

$$\begin{aligned} a &= e^{z_1\beta} + e^{z_2\beta} + e^{z_3\beta} + e^{z_4\beta} + e^{z_5\beta} \\ b &= e^{z_1\beta} \\ c &= e^{z_2\beta} \end{aligned}$$

이러한 근사식에 착안하여 1번쨰 동점 데이터에 대한 부분가능도함수의 일반적인 근사식은 다음과 같다.

$$L_1(\beta) \approx \frac{\exp \left[\sum_{\ell \in R_i} e^{z_\ell \beta} \right]}{\prod_{j=1}^{d_1} \left[\sum_{\ell \in R_1} e^{z_\ell \beta} - \frac{j-1}{d_1} \sum_{\ell \in R_i} e^{z_\ell \beta} \right]}$$

만약 2개의 동점이 있다면, 분모에 두 관측값의 평균을 사용한다.

$$\left(\frac{r_1}{r_1 + r_2 + r_3 + r_4 + r_5} \right) \left(\frac{r_2}{0.5r_1 + 0.5r_2 + r_3 + r_4 + r_5} \right)$$

앞선 예의 경우,

- 2개의 동점이 있다면 가중치는 $(1, \frac{1}{2})$
- 3개의 동점이 있다면 가중치는 $(1, \frac{1}{2}, \frac{1}{3})$
- 4개의 동점이 있다면 가중치는 $(1, \frac{3}{4}, \frac{2}{4}, \frac{1}{4})$
- r 개의 동점이 있다면, 가중치는 $(1, \frac{r-1}{r}, \frac{r-2}{r}, \dots, \frac{1}{r})$

Part VIII

Performance Evaluation Metrics

본 Chapter의 지표들에 대한 카테고리는 추순규(2015)[6], 석진미(2017)[4], Ping Wang et al.(2017)[52], 임성빈(????)[5]을 참고하여 정리하였다.

임상 분야에서 질병의 유무를 예측함에 있어,
반응 변수(response variable)가 단순히 ”질병의 유/무” 구분을 의미하는 binary data이면

- Receive operating characteristic curve(**ROC Curve**) & Area under the curve(**AUC**)
- Net reclassification improvement(**NRI**)
- Mean Square Error(**MSE**) & Mean Absolute Error(**MAE**)
- Cox and Snell의 결정계수(R^2) & Nagelkerke의 결정계수(R^2)
- Brier Score
- Hosmer-Lemeshow Test

등을 사용할 수 있다.

하지만, 반응 변수(response variable)가 생존 시간일 경우에, 위 지표는 생존 자료가 가질 수 있는 정보를 고려하지 못하는 제약이 있으므로 그에 맞는 다음과 같은 평가 지표를 사용해야 한다.

- Ignoring time to event **C-index**)
- Chambliss and Diao의 **C statistic**)
- Gonen and Heller의 **K**)
- Harrell의 Concordance Index(**C-index**)
- Heagerty의 intergrated AUC
- Uno의 **C-index**
- Pencina and Uno의 **NRI**
- **Ctd Index**

다만, 반응 변수(response variable)가 binary data인 경우와, 생존 시간인 경우 각각 사용해야하는 평가 지표가 반드시 위와 같이 염밀하게 구분하여 사용되는 것은 아니고, 위 구분도 단지 survey 과정에서 편의를 위하여 임의로 구분한 것임을 염두에 두자. 어차피 앞으로 연구 목적에 따라 사용할 지표가 정해지게 될 것이며, 위 분류는 단지 참고용으로만 알고 있도록 하자.

18 반응 변수가 binary data인 경우

18.1 Receive operating characteristic curve(ROC Curve) & Area under the curve(AUC)

ROC curve는 민감도(sensitivity)와 특이도(specificity)를 이용해 구할 수 있다.

[민감도(sensitivity, true positive rate)] 실제 질병이 있는 대상을 검사 결과, 질병이 있다고 판단할 확률

[특이도(specificity, false negative rate)] 실제 질병이 없는 대상을 검사 결과, 질병이 없다고 판단할 확률

예측 모형을 통해 얻어진 질병이 있을 확률을 p 라고 할 때, p 값이 ”어느 정도”²⁷ 이상이면, 질병이 있다고 판단할 수 있다. 특히, 특이도보다 민감도가 더 중요한 상황에서는 p 가 낮아도 질병이 있다고 판단하게 된다.

ROC curve는 질병이 있다고 판단할 확률 p 를 이동하면서, 민감도와 특이도를 이용하여 구할 수 있다.

예를 들어, $p = 0.1$ 을 기준으로 질병의 유무를 판단한다고 하면, 그 때의 민감도와 특이도를 구하고 y 축을 민감도, x 축을 (1-특이도)를 이용하여 구할 수 있다.

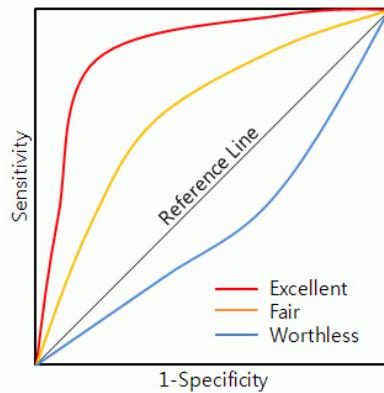


Figure 154: ROC curve

그림 154에서 Excellent, Fair, Worthless는 단지 참고용이며, 어느 질병이나에 따라 기준이 달라질 수 있음을 염두에 두자.

AUC는 ROC curve의 아래 면적을 말하며, 0에서 1 사이의 값을 갖는다. 1에 가까울수록 모형의 예측력이 뛰어나다고 볼 수 있고, 1에 가까워진다는 것을 그림 154에 벗어 표현하면, ROC curve가 reference line에서 빨간색 선 방향에 가까워진다는 것을 의미한다.

Note 28. AUC를 이용하여 두 모형을 비교할 경우, 통계적 유의성 검정

모형이 2개가 있고, 이 두 모형을 비교하는 경우, AUC가 1에 더 가까운 모형이 예측력이 뛰어나다고 평가한다.

또한, Henley and McNeil(1983)[32]에 따르면, 이 두 모형의 예측력 차이가 통계적으로 유의한지를 평가할 수 있는데, 첫 번째 모형을 Model₁, 두 번째 모형을 Model₂, 그리고 각 모형에서 얻어진 AUC를 AUC₁, AUC₂라고 하였을 때, 다음과 같은 관계가 있다.

$$\frac{AUC_1 - AUC_2}{SE[AUC_1 - AUC_2]} \sim Z(0, 1)$$

표준오차에 대한 식은 다음과 같다.

$$\begin{cases} \text{두 모형이 독립일 경우} & SE[AUC_1 - AUC_2] = \sqrt{SE^2(AUC_1) + SE^2(AUC_2)} \\ \text{두 모형이 독립이 아닐 경우} & SE[AUC_1 - AUC_2] = \sqrt{SE^2(AUC_1) + SE^2(AUC_2) - 2rSE^2(AUC_1)SE(AUC_2)} \end{cases}$$

단, r 은 두 AUC의 상관성을 의미한다.

²⁷”어느 정도”의 기준은, 상황에 따라 달라지게 된다. 어느 질병은 0.1 이상이어도 질병이 있다고 판단하게 되지만, 어느 질병은 0.7 이상이어야 질병이 있다고 판단하게 되기도 한다.

18.2 Net reclassification improvement(NRI)

NRI는 새로운 factor가 기존 모형에 추가되었을 때, 예측력 향상을 평가하기 위한 목적으로 제안되었다.

따라서, 이 방법은, 하나의 모형이 다른 모형에 nested되어있는 경우, 즉, 두 모형이 독립이 아닌 경우에 대해서 예측력을 비교할 때 사용된다.

기존 예측모형과, 새로운 인자가 포함된 예측모형을 각각 Model₁, Model₂라고 하면, 각 예측모형의 로지스틱 회귀모형은 다음과 같다.

$$\text{Model}_1 = \log \left(\frac{p_1(Y=1)}{1 - P_1(Y=1)} \right) = \alpha_1 + \beta_{(1,1)}x_1 + \cdots + \beta_{(n,1)}x_n$$

$$\text{Model}_2 = \log \left(\frac{p_2(Y=1)}{1 - P_2(Y=1)} \right) = \alpha_2 + \beta_{(1,2)}x_1 + \cdots + \beta_{(n,2)}x_n + \beta_{(n+1,2)}x_{n+1}$$

$Y = 1$ 인 경우를 질병이 있는 집단(event)이라고 할 때, Model₁, Model₂를 통해, event 집단이 될 예측 확률 p_1 , p_2 를 갖는다.

$$p_1(Y=1) = \frac{1}{1 + e^{-(\alpha_1 + \beta_{(1,1)}x_1 + \cdots + \beta_{(n,1)}x_n)}}$$

$$p_2(Y=1) = \frac{1}{1 + e^{-(\alpha_2 + \beta_{(1,2)}x_1 + \cdots + \beta_{(n,2)}x_n + \beta_{(n+1,2)}x_{n+1})}}$$

각 i 번째 대상은 모형에서의 예측 확률을 통해 event 집단에 속할 확률을 가지며, 어떤 cut point를 기준으로 event 집단에 속할 확률을 ”낮음, 중간, 높음”과 같이 구분할 수 있다. Model_k ($k = 1, 2$)에서 구해진 예측확률 p_k 를 cut point를 기준으로 분류표를 작성하는 것은 Note 29를 참고하자.

Note 29. Model_k ($k = 1, 2$)에서 구해진 예측확률 p_k 를 cut point를 기준으로 작성한 분류표와 이에 따른 NRI 계산

Table 73: 재분류표

Event		Model ₂			Non-event		Model ₂		
		낮음	중간	높음			L	M	H
Model ₁	낮음	a	b	c	Model ₁	낮음	j	k	l
	중간	d	e	f		중간	m	n	o
	높음	g	h	i		높음	p	q	r

Table 73은 cut point를 기준으로 하여, 범주를 총 3개 지점으로 잡은 경우의 재분류표를 나타낸다. NRI는 일반적으로 cut point를 기준으로 범주를 2 3개로 나누어 이용한다. 재분류표를 이용하여 기존 예측모형에 비해 새로운 인자가 포함된 예측 모형이 event 집단과 non-event 집단을 더 잘 분류했는지를 판단한다.

분류표를 이용한 NRI의 정의는 다음과 같다.

$$\text{NRI} = [P(\text{up}|Y=1) - P(\text{down}|Y=1)] - [P(\text{up}|Y=0) - P(\text{down}|Y=0)]$$

$P(\text{up}|Y=1)$ 은 event 집단에서 Model₂의 예측 확률이 더 높게 분류된 확률을 의미한다.

반대로 $P(\text{down}|Y=1)$ 은 event 집단에서 Model₂의 예측 확률이 더 낮게 분류된 확률을 의미한다.

$$\hat{P}(\text{up}|Y=1) = \frac{b+c+f}{a+b+c+d+e+f+g+h+i} \quad \hat{P}(\text{down}|Y=1) = \frac{d+g+h}{a+b+c+d+e+f+g+h+i}$$

$$\hat{P}(\text{up}|Y=0) = \frac{d+g+h}{j+k+l+m+n+o+p+q+r} \quad \hat{P}(\text{down}|Y=0) = \frac{m+p+q}{j+k+l+m+n+o+p+q+r}$$

추정된 확률들을 이용하여 다음과 같이 NRI를 나타낼 수 있다.

$$\begin{aligned} \hat{\text{NRI}} &= [\hat{P}(\text{up}|Y=1) - \hat{P}(\text{up}|Y=0)] + [\hat{P}(\text{up}|Y=0) - \hat{P}(\text{up}|Y=0)] \\ &= \left[\left(\frac{b+c+f}{n_{\text{event}}} \right) - \left(\frac{d+g+h}{n_{\text{event}}} \right) \right] + \left[\left(\frac{m+p+q}{n_{\text{non-event}}} \right) - \left(\frac{k+l+o}{n_{\text{non-event}}} \right) \right] \end{aligned}$$

새로운 예측모형의 예측력이 더 뛰어나다면, NRI의 값은 0보다 커질 것이다. NRI 값이 0보다 큰 값을 갖는지 검정하기 위한 검정통계량은 다음과 같다.[49]

$$z = \frac{\hat{\text{NRI}}}{\hat{SE}(\text{NRI})}$$

$$\text{where } \hat{SE}(\text{NRI}) = \sqrt{\frac{\hat{P}(\text{up}|Y=0) - \hat{P}(\text{up}|Y=0)}{n_{\text{event}}} + \frac{\hat{P}(\text{up}|Y=1) - \hat{P}(\text{up}|Y=1)}{n_{\text{non-event}}}}$$

한편, continuous NRI(혹은 category-free NRI)도 있으며, 이는 반응변수에 절대적인 범주의 수가 존재하지 않는다고 볼 수 있을 때 사용한다.

사실 절대적인 범주의 수가 존재한다고는 할 수 없거나, 범주를 나눌 때 cut point가 논쟁이 될 수 있는데, 이 문제를 해결하기 위해 Pencina et al.(2011)[50]이 범주를 나누지 않는 방법을 제안하였다. 이 방법은, 기존 예측모형보다 새로운 예측모형이 예측 확률이 높으면 up, 낮으면 down으로 분류한다. 범주를 나누지 않기 때문에 continuous NRI 혹은 category-free NRI라고 이름이 붙었으며, cNRI라고 간략하게 표기한다.

$$cNRI = E[\text{sign}(p_2 - p_1 | Y = 1)] - E[-\text{sign}(p_2 - p_1 | Y = 0)]$$

p_1, p_2 는 각각 기존 예측모형과 새로운 예측모형으로부터 얻어진 예측확률을 의미한다. event 집단과 non-event 집단의 독립을 가정하면, cNRI의 검정통계량은 다음과 같이 나타낼 수 있다.

$$z_{cNRI} = \frac{\hat{NRI}}{4 \left[\frac{\hat{P}(\text{up}|\text{event}) - \{1 - \hat{P}(\text{up}|\text{event})\}}{n_{\text{event}}} + \frac{\hat{P}(\text{up}|\text{non-event}) - \{1 - \hat{P}(\text{up}|\text{non-event})\}}{n_{\text{non-event}}} \right]}$$

검정통계량을 통해 유의한 결과를 얻게 되면, 새로운 예측모형과 기존 예측모형의 예측력에 유의한 차이가 있다고 할 수 있다.

18.3 Mean Square Error(MSE) & Mean Absolute Error(MAE)

작성중...

18.4 Cox and Snell의 결정계수(R^2) & Nagelkerke의 결정계수(R^2)

주어진 자료에서 구축된 특정 모형이 설명할 수 있는 종합적인 척도로, 모형이 설명하는 변동량(explained variation, R^2)을 사용할 수 있다. 즉, R^2 를 사용하여, 결과변수에 대한 독립변수의 영향력을 확인할 수 있으며, 해당 값이 클수록 모형에 사용된 독립변수들의 결과변수에 대한 설명력이 높다고 할 수 있다.

작성중...

18.5 Brier Score

Brier score는, 로지스틱 회귀모형에서 Nagelkerke의 결정계수와 함께 종합적 모형 수행력 척도로 많이 사용되는 지표이다. 각 관찰 값에 대해 실제 자료 y 와 예측 결과 \hat{p} 간의 차이를 식 19과 같이 계산한 뒤, 이들을 평균한 값을 의미한다.

$$y^*(1 - \hat{p})^2 + (1 - y)^*\hat{p}^2 \quad (19)$$

Brier score는 0에서 1 사이의 값을 가지며, y 와 \hat{p} 의 결과가 일치할 수록 0, 일치하지 않을수록 1에 가까워진다.

Brier score는 자료 y 의 발생률에 의존하며, 발생률이 작으면 brier score의 최대값도 작아진다는 단점이 있다.

이러한 단점을 보완하기 위해 scaled Brier score도 제안되었다[61]. Scaled Brier score는 식 19의 최대값을 사용하여, 다음 식 20과 같이 0부터 100 사이의 값을 가지도록 정의된다. 값이 클수록 모형의 적합도가 좋음을 의미한다.

$$1 - \frac{\text{Brier Score}}{\max[\text{Brier Score}]} \quad (20)$$

18.6 Hosmer-Lemeshow Test

Hosmer-Lemeshow 검정은 로지스틱 회귀모형으로부터 예측 확률을 계산한 후, 이를 10개의 quantiles로 나눈 후, 각 그룹 내에서 기대 빈도와 관찰 빈도 간의 교차표를 대상으로 χ^2 검정을 실시하는 것이다.

만일 검정 결과가 유의하지 않다면, 모형의 적합도가 높음을 의미한다.

Hosmer-Lemeshow 검정의 단점은, 표본 수가 커지는 경우, 관찰 값과 예측 값 사이에 유의한 차이가, 실제로는 없음에도 불구하고 마치 존재하는 것으로 평가되는 제1종 오류가 증가한다는 점이다.

19 반응 변수가 생존 시간인 경우

19.1 Ignoring time to event (logistic C)

이 방법은 censored 부분을 그냥 버리고, event가 발생한 survival time을 이용하여 ROC curve를 구하는 방법이다. 당연히 censored가 많을 때에는 적용하기 어려워진다.

19.2 Harrell의 Concordance Index(C-index)

Harrell의 C-index는 binary 종속 변수에서 Hanley and McNeil(1982)[31]의 C-index를 구하는 비모수적인 방법을 생존 자료에 적용시킨 방법이다.

생존 시간을 \hat{T}_i , 중도절단 시간을 C_i 라고 정의하면, 관찰된 생존 시간은 $T_i = \min(\hat{T}_i, \delta_i)$ 로 정의할 수 있고, 중도절단 여부는 다음과 같다고 하자. (식 2의 리마인더이다.)

$$\begin{aligned} & \{(T_i, \delta_i), i = 1, \dots, n\} \\ & \text{단, } T_i = \min(\tilde{T}_i, \delta_i) \quad \text{and} \quad \delta_i = I(\tilde{T}_i < C_i) \\ & \text{즉, } \delta_i = \begin{cases} 1 & \text{if } \hat{T}_i < C_i \text{ (즉, 관측된 경우)} \\ 0 & \text{if } \hat{T}_i \geq C_i \text{ (즉, right censored만 고려한다는 가정 하에 중도절단된 경우)} \end{cases} \end{aligned}$$

Harrell의 C-index를 정의하기 위해서는, 비교 가능한 쌍과 비교 불가능한 쌍을 정의해야 한다.

- i 번째 대상과 j 번째 대상이 모두 중도절단된 경우이거나, 모두 중도절단되지 않은 경우에 생존 시간의 비교가 가능하다.
- i 번째 대상과 j 번째 대상 둘 중 하나만 중도절단된 경우는 2가지 경우로 나뉘게 된다.
 - $T_i < T_j, \delta_i = 0$ 또는 $T_i > T_j, \delta_i = 0$ 인 경우는 생존 시간 비교가 불가능하다.
 - $T_i < T_j, \delta_i = 1$ 또는 $T_i > T_j, \delta_i = 1$ 인 경우는 생존 시간 비교가 가능하다.

비교 가능한 쌍에서

- $T_i < T_j, \delta_i = 1$ 일 경우 점수를 1
- $T_i > T_j, \delta_i = 1$ 일 경우 점수를 0

으로 두면, Harrell의 C-index는 다음과 같이 정의된다.

$$C_{\text{Harrell}} = \frac{\text{비교 가능한 쌍에서의 점수의 합}}{\text{비교 가능한 쌍의 수}}$$

조금 다르게 표현하면,

$$T_c = P(r(X_i) > r(X_j) \text{ and } T_i > T_j) + P(r(X_i) > r(X_j) \text{ and } T_i < T_j)$$

$$T_d = P(r(X_i) > r(X_j) \text{ and } T_i < T_j) + P(r(X_i) > r(X_j) \text{ and } T_i > T_j)$$

여기서, $r(X_k) = \text{risk score}$

라고 하면

$$C = \frac{T_c}{T_c + T_d}$$

C-index는 0.5에서 1 사이의 값을 가지며, 1에 가까워질수록 model discrimination이 잘 되는 것이라고 이야기한다.

만약, 다른고자 하는 데이터에 중도절단된 대상이 없다면, 모든 쌍들이 비교 가능한 쌍이 되며, C-index는 Mann-Whitney Statistics와 같아지게 된다.[44]

Note 30. C-index를 이용하여 두 모형을 비교할 경우, 통계적 유의성 검정

기존 모형의 C-index와, 새로운 factor가 추가된 모형의 C-index를 각각 구하여, 새로운 모형에서의 예측력이 더 향상되었는지 판단하기 위해서는, 두 C-index의 차이가 0인지의 여부를 검정하면 된다.

검정을 위해서는 두 모형에서 얻어진 C-index의 차이에 대한 분산을 구하여 t검정을 진행할 수 있다.

그러나 두 모형을 독립이라고 간주할 수 없기 때문에, C-index의 차이에 대한 분산을 구하기가 쉽지 않다.

따라서 Bootstrap 방법을 이용하여, 두 C-index의 차이에 대한 신뢰구간을 구하여, 95% 신뢰구간에 0이 포함되는지 여부로 예측력 향상을 판단할 수 있을 것이다.

Note 31. C-index의 95% 신뢰구간 구하기

1. Nam's Method 작성중... (참고문헌 : [55])

2. Modified Kendall's τ

3. Simplified Method

작성중...

19.3 Heagerty의 intergrated AUC

Heagerty and Zheng(2005)[33]는 x 축을 시점 t , y 축을 시점 t 에서의 ROC curve의 아래 면적인 AUC 값으로 갖는 그래프를 그려 그 아래 면적을 적분하여 iAUC(integrated AUC) 값을 구하였다.

ROC curve는 모든 가능한 지점에서 민감도(sensitivity)와 특이도(specificity)의 관계를 표현한 곡선이다. 어떤 시점 t 에서의 민감도와 특이도는 다음과 같이 정의된다.

$$\text{sensitivity}^I(c, t) = P(M_i > c | T_i = t) = P(M_i > c | dN_i^*(t) = 1)$$

$$\text{specificity}^D(c, t) = P(M_i \leq c | T_i > t) = P(M_i \leq c | dN_i^*(t) = 0)$$

위 식에서, M_i 는 공변량 벡터와 회귀계수 벡터의 곱인 $\beta'x$ 를 의미한다. 또한 $dN_i^*(t) = I(T_i \leq t) - I(T_i \leq t-)$ 를 나타내며, 시점 t 에서 사건이 발생한 경우 1, 아닌 경우 0을 의미한다.

시간 T 에 따른 ROC curve를 시간 종속 ROC curve라 하며, 작성중...

Part IX

Results of Survey

[Neural Network with Life Data]

임성빈(????)[5]에 따르면, David Faraggi and Richard Simon(1995)[23]의 연구가 'Survival Data에 적용된 최초의 신경망 모형'이라고 소개되고 있다.

다만, Ping Wang et al.[52]에 따르면, David Faraggi and Richard Simon의 전후에 출판된 페이퍼들도 소개되고 있으며, 내가 직접 찾은 페이퍼도 있다. 이들도 함께 살펴보고자 한다.

[Neural Network with Image Data]

또한, 위 연구들은 임상 데이터나 유전자 데이터에 집중하고 있으나, 우리는 이미지 데이터도 고려하고 있다.

이미지 데이터를 이용한 Survival Prediction을 연구한 페이퍼 목록을 만들 수 있었다.

다만, 관련 연구가 많이 있을 것으로 기대되지만, 기대만큼 많은 시간을 투입하여 만든 목록은 아니므로, 조금 더 찾아볼 필요는 있다.

[Feature Extraction of Image Data]

한편, 이미지 데이터를 넣는다고 해도, 어떠한 이미지를 넣어야하는지가 관건이다. 이에 임성빈 박사는 이미지 데이터의 feature extraction에 대한 고민을 하고 있었다.

나 또한 이에 공감하고 있으며, 현재는 이미지에 들어있는 주요 object에 대한 patch를 뽑아내는 방안을 생각해 보았다.

* 가급적이면 본 노트의 notation에 맞춰서 수정하고자 하였지만, 여전히 페이퍼의 notation을 따를 수도 있다. 이를 감안해서 보도록 하자.

20 Neural Network with Life Data

20.1 A practical application of neural network analysis for predicting outcome

분포 가정 없이 alive/death 상태 추정이 이루어진다는 점에서 **non-parametric approach**라고 할 수 있다. Kaplan-Meier과 성능 비교를 진행한다.

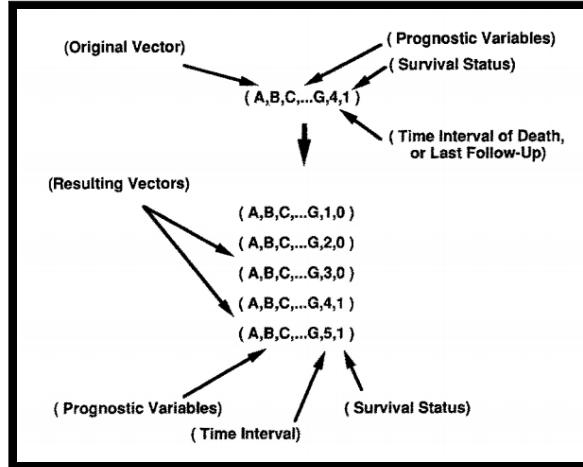


Figure 155: 데이터 전처리

Data vector를 $(A, B, C, D, E, F, G, T, S)$ 로 정의하며, 이들을 전처리하는 과정은 Figure 155와 같이 표현되어 있다. 여기서,

$$T = \begin{cases} 1 & 90 \text{ percent at 12 months} \\ 2 & 80 \text{ percent at 18 months} \\ 3 & 70 \text{ percent at 27 months} \\ 4 & 60 \text{ percent at 40 months} \\ 5 & 50 \text{ percent at 60 months} \end{cases}$$

$$S = \begin{cases} 0 & \text{if surviving at maximum follow up} \\ 1 & \text{if dead with maximum follow up} \end{cases}$$

만일 $S = 0$ 이면, T 는 마지막으로 follow-up한 시간이 되며, $S = 1$ 이면 T 는 사망이 관측된 시간을 의미한다.

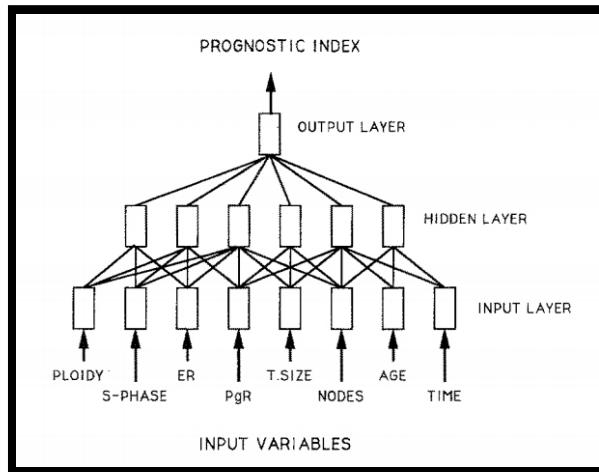


Figure 156: 신경망 모형

이렇게 전처리한 데이터를 Figure 156과 같이 적합하였다.

20.1.1 의견

알려진 생존분석 방법이 아닌, 순수한 neural network 방법에 life data를 적용한 방법이다. Neural network 자체만으로는 censored data에 대한 control이 된다는 보장이 없음에도 이를 검증하지 않고 적용했다는 비판을 할 수 있다.

20.2 Survival analysis and neural nets

이 연구에서는 우선 feed forward neural network를 hidden layer가 없는 경우(one-layer network)와 hidden layer가 1개 있는 경우(two-layer) 두 경우를 소개하고 있다. 이는 이후에 나올 **regression models for survival data as neural net**과 관련이 깊으므로, 이 노트에도 실어놓았다.

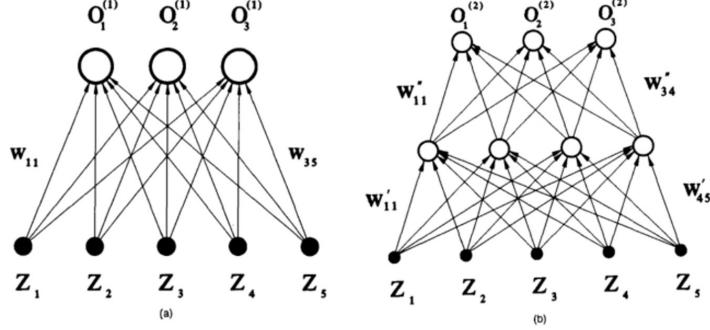


Figure 157: Liestbl et al.(1994)에서 소개하는 One-layer network와 Two-layer network

1. One-layer Network

$$O_k^{(1)}(z; w) = g \left(\sum_j w_{kj} z_j + w_{k0} \right) \quad (21)$$

여기서,

- $z = (z_1, z_2, \dots, z_p)$: Covariates
 - $w_{k\ell}$: Weights for the connection from input node j to output node k
- 이에 따른 cost function은 $E^{(1)}$ 으로 정의된다.

$$E^{(1)} = \sum_{\ell} \sum_k [O_k(z_{\ell}; w) - Y_{k\ell}]^2$$

2. Two-layer Network

$$O_k^{(2)}(z; w) = g \left[\sum_{\ell} w''_{k\ell} h \left(\sum_j w'_{\ell j} z_j + w'_{\ell 0} \right) + w''_{k0} \right]$$

이에 따른 cost function은 $E^{(1)}$ 으로 정의된다.

$$E^{(2)} = - \sum_{\ell} \sum_k [Y_{k\ell} \log O_k(z_{\ell}; w) + (1 - Y_{k\ell}) \log(1 - O_k(z_{\ell}; w))]$$

o] 페이퍼에서는 위의 cost function을 최소화하는 w 를 찾는 것이 목표라고 서술하고 있다.

3. 그리고 regression models for survival data as neural net을 다음과 같이 정의하고 있다.

우선, hazard function이 $\lambda(t)$ 이고, 반응 변수인 survival time이 T 일 때,

$$\lambda(t) = \frac{T < t + \delta t | T \geq t}{\delta t}$$

그러면 Cox 비례위험모형은 다음과 같이 서술할 수 있다.

$$\lambda(t) = \lambda_0(t)e^{\beta^T z}$$

여기서,

- $z = (z_1, \dots, z_p)^T$: Covariates
- $\lambda_0(t)$: 알려지지 않은 baseline hazard for individuals with $z = 0$
- $\beta = (\beta_1, \dots, \beta_p)^T$: regression coefficients

T 가 disjoint intervals I_k ($k = 1, \dots, m$)을 이용하여 그룹을 나누었다고 하자.

이 때 I_k 는 $t_{k-1} \leq t < t_k$ ($0 = t_0 < t_1 < \dots < t_m < \infty$)로 표시할 수 있으며, q_k 는 다음과 같은 조건부 생존률(conditional survival probabilities)로 나타낸다.

$$q_k = P(T \geq t_k | T \geq t_{k-1})$$

2개의 그룹으로 나누어진 경우, 다음과 같은 Cox 비례위험 모형을 고려할 수 있다.

$$\frac{p_k(z)}{q_k(z)} = \frac{p_k(0)}{q_k(0)} e^{\beta^T z} = e^{\beta^T z + \theta_k} \quad (22)$$

$$\text{여기서, } p_k = 1 - q_k, \quad \theta_k = \log \left(\frac{p_k(0)}{q_k(0)} \right)$$

Prentice and Gloeckler(1978)[53]와 Kalbfleisch and Prentice(1980)[42]의 방법에 따라, 식 (22)이 다음과 같다.

$$-\log(q_k(z)) = \Lambda_{0k} e^{\beta^T z} = e^{\beta^T z + \theta_k}$$

$$\text{여기서, } \Lambda_{0k} = e^{\theta_k} = \int_{t_{k-1}}^{t_k} \lambda_0(t) dt$$

따라서,

$$E = \sum_{i=1}^n \sum_{k=1}^{m_i} [D_{ki} \log p_k(z_i) + (1 - D_{ki} \log q_k(z_i))]$$

여기서,

- D_{ki} : the indicator for individual i dying in the interval I_k
- $m_i \leq m$: the number of intervals in which individual i is observed

이제 여기에 neural net을 붙이기 위해, 다음과 같은 가정을 하였다.

$$w_{1j} = w_{2j} = \dots = w_{mj} = \beta_j \quad (j = 1, \dots, p)$$

$\beta = (\beta_1, \dots, \beta_p)^T$, $\theta_k = w_{k0}$ 라고 하자. 그러면 식 (21)를 다음과 같이 출력할 수 있다.

$$O_k^{(1)}(z; \beta, \theta_k) = g(\beta^T z + \theta_k)$$

20.2.1 의견

이 연구에서는 위험이 일정한 것으로 가정하고(즉, 수명 함수가 지수 분포를 따르는 것으로 가정) 출력 노드가 각 간격마다 설정된 시간 간격으로 그룹화하였다. Hidden layer가 없고 동일한 입력 노드에서 모든 출력 노드까지 동일한 가중치 조건에서 neural network는, 각 개인의 조건부 이벤트 확률을 예측하도록 학습되며, 그 결과는 grouped Cox 비례위험모형 버전과 동일하다.

동일한 데이터의 시간에 따른 ”일정하지 않은 위험“은, 각 출력 노드에 대한 가중치가 다를 때까지 모델링될 수 있다. Hidden layer를 추가하면, 비선형 공변량 효과가 있는 Cox neural network라는 하이브리드 형식으로 만들어진다. 비선형성 정도는 Hidden layer 수와 activation function의 선택에 달려 있다.

20.3 A Neural Network Model for Survival Data

여기에서 또한 Cox 비례위험모형에 neural network를 적용하고 있다.

1. Neural Network

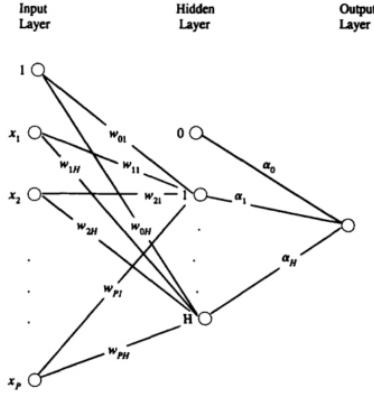


Figure 158: Single hidden layer neural network to single output

$$g(x_i, \theta) = \alpha_0 + \sum_{h=1}^H \alpha_h f(w'_h x_i) = \alpha_0 + \sum_{h=1}^H \frac{\alpha_h}{1 + e^{-w'_h x_i}}$$

여기서,

- Covariates: $x_i = (x_{i0}, x_{i1}, \dots, x_{iP})'$ ($i = 1, \dots, n$), ($p = 0, \dots, P$)
- Squashing function: $f()$
- Hidden node: $h = 1, \dots, H$
- The vector of weights: $w_h = (w_{0h}, w_{1h}, \dots, w'_{Ph})$
- The vector of unknown parameters:²⁸

$$\theta = (w_{01}, w_{11}, \dots, w_{p1}, w_{02}, \\ w_{12}, \dots, w_{p2}, \dots, w_{0H}, \\ w_{1H}, \dots, w_{pH}, \\ \alpha_0, \alpha_1, \dots, \alpha_H)'$$

2. Neural Network Survival Models

Hazard function의 다음과 같다.

$$h(t, x_i) = h_0(t) e^{\beta x_i}$$

파라미터의 벡터인 β 는 다음과 같은 부분 가능도 함수(maximizing the partial likelihood function)를 최대화하여 추정할 수 있다.

$$L_c(\beta) = \prod_{i \in w} \frac{e^{\beta x_i}}{\sum_{j \in R_i} e^{\beta x_j}}$$

여기에서는 \circ partial likelihood를 network $g(x_i, \theta)$ 로 바꾸었다. 즉 Hazard function의 다시 다음과 같이 정의된다.

$$h(t, x_i) = h_0(t) e^{g(x_i, \theta)}$$

또한, 추정해야 할 부분 가능도 함수(partial likelihood function)도 다음과 같이 수정된다.

$$L_c(\theta) = \prod_{i \in w} \frac{\exp \left[\sum_{h=1}^H \frac{\alpha_h}{1 + e^{-w'_h x_i}} \right]}{\sum_{j \in R_i} \exp \left[\sum_{h=1}^H \frac{\alpha_h}{1 + e^{-w'_h x_j}} \right]}$$

부분 최대 가능도 추정량을 구하기 위해 Newton-Raphson Method를 사용하였다.

이러한 방법은 가속수명모형(the accelerated failure time model), Buckley-James model에도 적용할 수 있다.

20.3.1 의견

Cox 비례위험모형에 covariate의 일반적인 선형 조합 대신, 비선형 함수를 혼용하도록 한 방법이다. 이 방법은 Cox 모델의 비례 위험 요소를 유지하지만, 단순한 Cox 모델에서 누락될 수 있는 입력 데이터의 복잡성과 interaction을 모델링하는 기능을 제공한다.

²⁸이 파라미터의 개수는 다음과 같이 계산할 수 있다. $m = (H + 1) + (P + 1)H$

20.4 On the use of artificial neural networks for the analysis of survival data

여기에서는 Choong et al.(1993)과 deSilva et al.(1994)의 방법을 follow up하고, 이들의 방법을 적용할 시 bias가 발생한다는 점을 개선하기 위한 새로운 접근을 시도하고 있다.

Kaplan-Meier와 Cox 비례위험모형에 neural network를 붙이는 방식이다.

1. Some Results from Survival Statistics

- Survival Function:

$$S(t) = P(T \geq t) = e^{-\int_0^t h(t') dt'} \quad (23)$$

* $S(0) = 1, S(\infty) = 0$

- Hazard function:

$$h(t) = \lim_{\Delta t \rightarrow 0+} \frac{P(T \leq T < t + \Delta t | T \geq t)}{\Delta t}$$

- 생존 곡선은 Kaplan-Meier 방법에 기반한 최대 가능도 추정법(maximum likelihood approach)으로 추정될 수 있다.

$$\hat{S}(t) = \prod_{k|t_k < t} \left(\frac{n_k - d_k}{n_k} \right)$$

여기서,

– d_k : The number of subjects that fail at time t_k .

– n_k : The total number of subjects that fail or are censored at time t_k or later.

- x 를 입력 벡터(a vector of inputs), β 를 파라메터 벡터(a vector of parameters)라고 하고, Cox 비례위험모형에 기반한 생존 함수(survival function), 위험 함수(hazard function)를 다음과 같이 정의할 수 있다.

$$S(t; x) = S_0(t)^{e^{\beta^T x}}$$

○] 때, $S_0(t) = e^{-\int_0^t h_0(t') dt'}$

$$h(t; x) = h_0(t) e^{\beta^T x}$$

○] 때, $h_0(t)$: arbitrary baseline hazard function

2. Previous ANN Approaches

$$E = \frac{1}{2} \sum_{i=1}^m (t_i - t_{out})^2$$

$$\implies t_{out} = \sum_{i=1}^m \frac{t_i}{m}$$

여기서,

- t_i : the training set outputs
- m : the sample size

Choong et al.(1993)과 deSilva et al.(1994)는, artificial neural network(ANN)를 사용하여, 피부암과 유방암에 따른 사망률(skin and breast cancer mortality)을 연구했다. 그들은 failure time과 censoring time을 사용하여 네트워크를 훈련시켰다고 한다. 그러나 ANN을 사용하여 중도절단된 시간보다 긴 생존 시간을 예측할 때, 예측 오류가 0으로 간주되고, 네트워크 가중치가 업데이트 되지 않는 문제가 있었다고 한다. 이는 censored data에 의해 도입되는 bias를 다소 제거하는 효과를 주겠지만, 여전히 남아있는 bias의 크기를 평가할 방법을 주지 않는다.

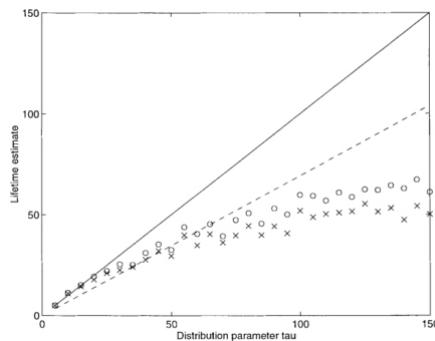


Figure 159: Choong et al.(1993)의 방법으로 추정된 observed failure의 평균(O 표시)와 censoring time의 평균(X 표시). 실선은 true mean이고 점선은 true median이다.

이 페이퍼에서는 위 두 페이퍼에서 논의된 bias의 효과를 보다 명확하게 보기 위해, 수명 분포가 지수분포를 갖는 집단으로부터 시뮬레이션을 수행했다.²⁹

$$S(t) = e^{-\frac{t}{\tau}} \quad (\text{여기서, } \tau > 0)$$

이 τ 값을 5부터 150까지 5씩 증가시키면서, 각각의 값마다 100번의 censoring time이 무작위로 배치된다. 또한 경과 시간이 100에 도달하기 전에 failed되지 않거나(not failed) censored된 객체는 censored로 간주되었다.

시뮬레이션의 결과는 Figure 159에서 확인할 수 있다.

- $\tau < 30$ 일 때 ANN는 비교적 평균을 잘 추정하는 모습을 볼 수 있다.
- $30 < \tau < 60$ 일 때, 추정 결과는 평균과는 거리가 멀어졌지만, 중앙값과는 가까운 모습을 볼 수 있다.
- $60 > \tau$ 일 때, 추정 결과가 평균과 중앙값 모두에 미치지 못하는 것을 볼 수 있다.

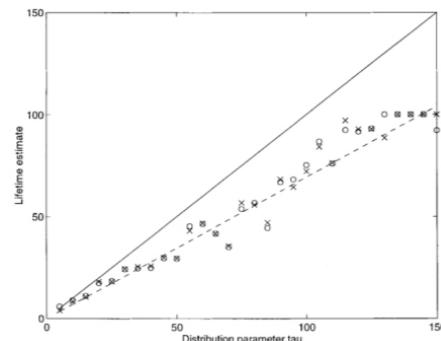


Figure 160: Kaplan-Meier 방법으로 추정된 observed failure의 평균(O 표시)와 censoring time의 평균(X 표시). 실선은 true mean이고 점선은 true median이다.

한편, 전통적인 non-parametric 방법인 Kaplan-Meier 방법으로 추정할 경우, Figure 160와 같이 추정량이 median을 비교적 정확하게 추정하는 것을 볼 수 있다. Choong et al.(1993)으로 추정한 Figure 159와 비교되는 부분이다.

²⁹Chapter 14.11에서는 지수분포에 대한 생존함수를 $S(t) = e^{-\lambda t}$ 로 서술하고 있고, 별도로 $\lambda = \frac{1}{\theta}$ 인 경우에 대해서도 서술하고 있다. 또한 MLE의 성질에 따라 $\hat{\lambda}_{MLE} = \frac{1}{\theta_{MLE}}$ 라고 할 수 있다. 이 페이퍼에서는 notation을 θ 대신 τ 로 기술한 것으로 보면 된다.

3. A New Approach

이 논문에서는 새로운 접근법으로, 객체의 수명을 추정하는 대신에, 객체의 $S(t)$, 즉 생존 함수를 추정하고자 시도하였다.

$S(t)$ 는 ANN을 이용하여 hazard function을 계산한 뒤, 이를 $S(t)$ 와의 관계를 이용하여 구할 수 있다.³⁰

식 (23)의 discretized version은 다음 식 (24)과 같이 표현할 수 있다.

$$\begin{aligned}\tilde{S}(t_j) &= e^{-\sum_{k=1}^j \tilde{h}_k} \\ &= \prod_{k=1}^j (1 - h_k) \\ &= \tilde{S}(t_{j-1})(1 - h_j)\end{aligned}\quad (24)$$

이 때,

- $\tilde{h}_k \geq 0$
- $h_k = 1 - e^{-\tilde{h}_j}$
- $\tilde{S}(0) = 1$

이제 i 번째 객체 각각에 대해, 경험적 생존 함수(empirical survival function)³¹ $S_i(t)$ 를 고려해보자.

t_f 가 어느 객체가 fail이 일어난 시간이라면,

$$S_i(t) = \begin{cases} 1 & t \leq t_f \\ 0 & t > t_f \end{cases} \quad (25)$$

식 (24)의 \tilde{S} 와 비교한다면, 식 (25)은 다음을 만족하는 경험적 위험률 요소(empirical hazard components) h_j^i 를 요구한다는 것이다.

- $j < j_{\text{crit}}$ must be zero
- One of $j = j_{\text{crit}}$ must be zero
where j_{crit} is the smallest value of j such that $t_f < t_j$

그리고 i 번째 객체 각각에 대해, t_c 가 중도절단시간이라고 할 때, 경험적 생존 함수는 다음과 같이 표시할 수 있다.

$$S_i(t) = \begin{cases} 1 & t \leq t_c \\ \text{unknown} & t > t_c \end{cases}$$

이 때 h_j^i 는 다음과 같이 요구된다.

- $j < j_{\text{crit}}$ must be zero
where j_{crit} is the smallest value of j such that $t_c < \frac{1}{2}(t_{j-1} + t_j)$

이제 추정된 위험률을 h_j 라고 할 때, 앞으로 최소화할 대상이 될 error term은 원래 다음과 같이 정의될 것이다.

$$E = \frac{1}{2} \sum_{i=1}^m (h_j^i - h_j)^2$$

여기서, m : sample size

이 논문에서는 이를 다음과 같이 확장하였다.

$$E = \left[\frac{1}{2} \sum_{k=1}^{n_f} (1 - h_j)^2 + \frac{1}{2} \sum_{k=1}^{n_c} (0 - h_j)^2 + \frac{1}{2} \sum_{k=1}^n (0 - h_j)^2 \right]$$

여기서,

- n_f is the number of subjects(객체) with failure times between $(j-1)\Delta t$ and $j\Delta t$
- n_c is the number of subjects(객체) that are censored between times $(j-\frac{1}{2})\Delta t$ and $j\Delta t$
- n is the number of subjects that have not failed or been censored by time $j\Delta t$

h_j 의 최소값은 $h_j = \frac{n_f}{n+n_f+n_c}$

20.4.1 의견

절단이 없는 부분에 대한 추정 결과와 중도절단된 부분에 대한 추정 결과를 분리해서 본 것이 적절하여 보였다.

데이터와 참고문헌을 구할 수 없어서, 재현할 수가 없는 상황이다.

³⁰생존 함수와 위험률 함수와의 관계는 Chapter 8를 참고하자.

³¹왜 ”경험적”이라는 말이 붙었는지는 아직 모르겠다.

20.5 Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach

여기에서는 partial logistic regression에 neural network를 붙이는 접근을 하였다.

1. Artificial Neural Networks and Generalized Regression Models

우선 feed forward ANN에 대해서 짚고 있다. Feed forward ANN은 non-linear multivariate regression method와 동치이다.

Node $h = 1, 2, \dots, H$, input node $j = 1, 2, \dots, J$, output node $k = 1, 2, \dots, K$ 라고 하고, observed input and observed responses를 y_{ik}^o 라고 표시하자. 모델에 의한 추정량(outputs)은 \hat{y}_{ik} 로 표기한다. 각 노드에서는 input x_{ij} 에 대한 weights w_{jh} 와 constant(bias) α_h 계산된다.

$$\hat{y}_k(x_i, w) = \phi_o \left(\alpha_k + \sum_{h=1}^H w_{hk} \phi_h \left(\alpha_h + \sum_{j=1}^J w_{jh} x_{ij} \right) \right)$$

Hidden node를 위한 activation function ϕ_h 는 다음과 같다.

$$\phi_h(u) = \frac{\exp(u)}{1 + \exp(u)}$$

GLM에서는 ϕ 를 link function이라고 부르며, 분포 가정에 따라 정의되는 형태가 다르다. 또한 이 link function은 위 activation function과 동치이다.

모델의 파라메터인 weights w 를 추정하기 위해 error function을 최소화하는 과정을 거친다. 가장 많이 사용되는 quadratic error는 다음과 같이 정의된다.

$$E = \sum_{k=1}^K \sum_{i=1}^n (\hat{y}_k(x_I, w) - y_{ki}^o)^2$$

다만, binary classification 문제를 풀기 위해서는, 다음과 같이 cross-entropy error를 이용한다.

$$E = - \sum_{k=1}^K \sum_{i=1}^n [y_{ki}^o \log \hat{y}_k(x_I, w) + (1 - y_{ki}^o) \log(1 - \hat{y}_k(x_I, w))] \quad (26)$$

Error function 식 (26)의 절대 최소값은, 는 K 개 output에 대한 n 개의 subjects에 대해 $y_{ik}^o = \hat{y}_{ik}$ 일 때 발생하며, 다음 식 (27)와 같이 표현된다.

$$E_{\min} = - \sum_{k=1}^K \sum_{i=1}^n [y_{ki}^o \log y_{ki}^o + (1 - y_{ki}^o) \log(1 - y_{ki}^o)] \quad (27)$$

2. Discrete Time Models for Survival Data

Continuous-time survival data일 때, hazard function $h(t)$ 는 다음과 같이 정의된다.

$$h(t) = \lim_{\Delta t \rightarrow 0} \frac{P(T < t + \Delta t)}{\Delta t}$$

다만 이 페이퍼에서는 discrete time을 고려하였다. 즉, L 개로 분할 된 $0 < t_1 < t_2 < \dots < t_L$ 을 고려하였다. 이는 continuous survival times $\ell = 1, 2, \dots, L$ 이 있을 때, $A_\ell = (t_{\ell-1}, T_\ell]$ 로 구분한 것이다. 이 때 $t_0 = 0$ 이고, ℓ_i 는 i 번째 subject에 있어 i 번째 시간의 interval 끝자락에서 식별된 것이다.

적절한 survival function을 다음과 같이 정의하고

$$S(t_\ell) = P(T > t_\ell)$$

discrete probability function을 다음과 같이 정의하면

$$f_\ell = P(T \in A_\ell) = S(t_{\ell-1}) - S(T_\ell) \quad (28)$$

discrete hazard rate h_ℓ 은 다음과 같은 conditional failure probability로 정의할 수 있다.

$$h_\ell = P(T \in A_\ell | T > t_{\ell-1}) = \frac{f_\ell}{S(t_{\ell-1})} \quad (29)$$

A_ℓ 을 무한대로 보내면, conditional failure probability h_ℓ 은 continuous hazard function $h(t)$ 에 수렴하게 된다. 식 (28)과 식 (29)로부터 다음 $S(t)$ 를 정의할 수 있다.

$$S(t) = \prod_{\ell: t_i \leq t} (1 - h_\ell)$$

i 번째 subject에 대한 가능도 함수에 대한 기여는, 시간 간격에 대한 conditional survival probability와 관심 이벤트가 발생하는 간격 A_ℓ 에서의 conditional failure probability의 곱으로 주어진다.

우중도절단만을 고려했을 때, uncensored subjects의 set U 가 기여하는 부분은

$$P(T_i \in A_{\ell_i}) = S(t_{\ell_i}) = \prod_{\ell=1}^{\ell_i-1} (1 - h_{i\ell})$$

censored subjects C 가 기여하는 부분은 다음과 같이 표현된다.

$$P(T_i > t_{\ell_i}) = S(t_{\ell_i}) = \prod_{\ell=1}^{\ell_i} (1 - h_{i\ell})$$

Censoring indicator를 $d_{i\ell} \circledast$ 라고 하고, $d_{i\ell} = 1$ 이면 uncensored subjects를, $d_{i\ell} = 0$ 이면 censored subject를 포함한다고 했을 때, 최종적으로 likelihood는 다음과 같이 표현할 수 있다.

$$L = \prod_{i=1}^n \prod_{\ell=1}^{\ell_i} h_{i\ell}^{d_{i\ell}} (1 - h_{i\ell})^{1-d_{i\ell}} = \prod_{\ell=1}^L \prod_{i \in R_i} h_{i\ell}^{d_{i\ell}} (1 - h_{i\ell})^{1-d_{i\ell}} \quad (30)$$

여기서, R_ℓ 은 the set of individuals at risk in the ℓ th interval of time. $R_\ell \circledast$ Bernoulli trial이라고 한다면, 식 (30)을 다음과 같이 고쳐쓸 수 있다.

$$L = \prod_{\ell=1}^L \binom{n_\ell}{s_\ell} h_\ell^{s_\ell} (1 - h_\ell)^{n_\ell - s_\ell}$$

여기서, n_ℓ 은 the number of subjects at risk, s_ℓ 은 the number of failures in the time interval ℓ .

이제 covariates를 고려해보자. David Cox는 grouped survival times를 위해 다음과 같은 proportional odds model을 제안하였다.

$$\frac{h_\ell(x_i)}{1 - h_\ell(x_i)} = \frac{h_\ell(0)}{1 - h_\ell(0)} \exp(\beta^T x_i)$$

Baseline hazard rate를 $\theta_\ell = \log \left(\frac{h_\ell(0)}{1 - h_\ell(0)} \right)$ 로 치환하면, 위 식으로부터 $h_\ell(x_i)$ 를 다음과 같이 표현할 수 있다.

$$h_\ell(x_i) = \frac{\exp(\theta_\ell + \beta^T x_i)}{1 + \exp(\theta_\ell + \beta^T x_i)}$$

3. Partial Logistic Regression Models with ANN (PLANN)

식 (30)에 log를 써우면 다음과 같다.

$$L = - \prod_{i=1}^n \prod_{\ell=1}^{\ell_i} [d_{i\ell} \log h_{i\ell} + (1 - d_{i\ell}) \log(1 - h_{i\ell})] \quad (31)$$

이와 같은 error function을 사용한다는 것은 hidden layer가 없고, logistic activation function ϕ_o 를 사용한 neural network를 사용한다는 것과 같다.

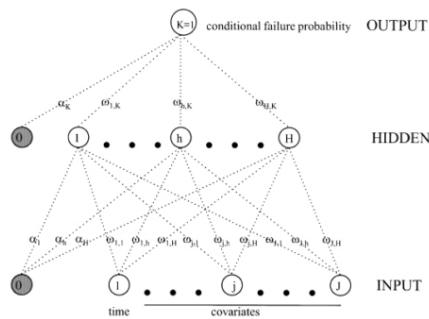


Figure 161: Feed forward neural network model for partial logistic regression(PLANN)

PLANN model은 하나의 input node j 에 설명변수 x_j 를 대입하고, 추가로 time interval a_ℓ input node를 추가한 것이다. 이는 Figure 3과 같은 그림으로 나타낼 수 있다. 결국 이 페이퍼는 Time interval node를 추가함으로써 time varying term을 control하겠다는 아이디어인 것이다.

설명변수가 categorical data라면, n_m 개 subjects와 s_m 개 events를 가진 공변량 값에 기초하여 $m = 1, 2, \dots, M$ 의 디자인 셀에 그룹화할 수 있다. 따라서 각 셀에 대한 empirical estimate \hat{h}_m 은 다음과 같이 얻을 수 있다.

$$\hat{h}_m = \frac{s_m}{n_m}$$

이러한 경우, error function을 다음과 같이 사용한다.

$$L = - \sum_{m=1}^M \left[\hat{h}_m \log \frac{h_i(x_i, a_i)}{\hat{h}_m} + (1 - \hat{h}_m) \log \frac{1 - h_i(x_i, a_i)}{1 - \hat{h}_m} \right] n_m \quad (32)$$

4. PLANN and Previously Proposed ANN Approaches for Grouped Survival Data

20.6 Comparison of the performance of neural network methods and Cox regression for censored survival data

20.6.1 데이터셋

여기서는 9가지 synthetic dataset를 생성하여 실험에 이용하였다.

Dataset 1. Anny Xiang et al.(2000)의 Synthetic Data

- Design 1: No censoring. $n = 100$, 50 replications.

$$\gamma = \exp [x_1 + 0.25x_2]$$

$$x_1 \sim \text{Bernoilli}(0.5) \quad x_2 \sim N(0, 1)$$

- Design 2: Average censoring for training and testing sets = 19% each. $n = 100$, 50 replications.

$$\gamma = \exp [x_1 + 0.25x_2]$$

$$x_1 \sim \text{Bernoilli}(0.5) \quad x_2 \sim N(0, 1)$$

- Design 3: Average censoring for training and testing sets = 20% each. $n = 100$, 50 replications.

$$\gamma = \exp [x_1 + 0.25x_2 + 0.2x_1x_2]$$

$$x_1 \sim \text{Bernoilli}(0.5) \quad x_2 \sim N(0, 1)$$

- Design 4: Average censoring for training and testing sets = 32%. $n = 100$, 50 replications.

$$\begin{aligned} \gamma = & \exp[2(x_1 + x_2) + 0.5x_3 + 1.0x_4 \\ & + 1.0(x_1x_2 + x_1x_3) \\ & + 0.5(x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4) \\ & + 0.5(x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 + x_1x_2x_3x_4)] \end{aligned}$$

$$x_1 \sim \text{Bernoilli}(0.25) \quad x_2 \sim \text{Bernoilli}(0.5) \quad x_3 \sim N(0, 1) \quad x_4 \sim N(0, 1)$$

- Design 5: Same as Design 4. Average censoring for training and testing sets = 70%. $n = 100$, 50 replications.

- Design 6: Average censoring for training and testing sets = 47%. $n = 100$, 50 replications.

$$\gamma = \exp [0.5(x_1 + x_2) + 0.25(x_3 + x_4) + 3.0(x_1x_2x_3 + x_1x_2x_4 + x_2x_3x_4)]$$

$$x_1 \sim \text{Bernoilli}(0.25) \quad x_2 \sim \text{Bernoilli}(0.5) \quad x_3 \sim N(0, 1) \quad x_4 \sim N(0, 1)$$

- Design 7: Non-proportional hazard function of Design 2. $\beta_1 = 1.0$ and $\beta_2 = 0.25$, x_1 and x_2 before the time point with 70% survival probability. Therefore. $\beta_1 = \beta_2 = 0$. $n = 100$, 50 replications.

- Design 8: Same as Design 3 except $n = 200$ cases.

- Design 9: Same as Design 5 except $n = 200$ cases.
-

20.6.2 요약

”Faraggi and Simon(1995)[23]의 방법”, ”Liestol et al.(1994)[43]의 방법”, 그리고 Buckley and James 방법(1979)[19]에 neural network를 붙인 ”수정된 Buckley and James 방법”을 비교하는 연구를 하였다.

1. Faraggi-Simon method Faraggi and Simon(1995)[23]의 방법은 Chapter 20.3에서 이미 정리하였다.
다시한 번 요약하자면, Faraggi and Simon(1995)는 Cox regression의 일반적인 공변량의 선형 결합 대신에, 비선형 함수를 허용하도록 일반화하였다.

2. Liestol-Andersen-Andersen method

Liestol et al.(1994)[43]의 방법은 Chapter 20.2에서 이미 이미 정리하였다.

다시한 번 요약하자면, Liestol et al.(1994)는, 생존 시간은, 위험이 일정하다는 가정 하에서, 출력 노드가 시간 간격으로 그룹화된다.

Hidden layer가 없고, 입력 노드에서 출력 노드까지 동일한 가중치 조건으로 설정된 NN은, 각 objects에 대한 conditional event probability를 training하도록 예측되며, 그 결과는 그룹화된 Cox regression과 동일하다.

각 출력 노드에 대한 가중치가 허용될 때는, 시간에 따른 일정하지 않은 위험을 모델링할 수 있다. Hidden layer를 추가하면, 비선형 covariate와 Cox-NN 하이브리드 형식의 모델이 만들어지고, linearity는 이 hidden layer의 수와 activation function에 달려 있다.

3. Modified Buckley-James method

선형 회귀 분석에 적용된 원래의 Buckley-James 방법의 경우, censored survival time은 공변량과 잔여 Kaplan-Keier로부터 추정된 회귀선에 따라 예상 값이 결정되게 된다. Residual distribution은 파라미터의 함수이기 때문에, 반복 추정을 하는 방법이 이용되며, 각 반복에서의 추정 값은, 현재의 파라미터의 추정값을 기반으로 한다.

Modified Buckley-James 방법의 경우, NN outputs은 fitted regression line 대신 사용되며, ... 작성중...

위 세 방법과 원래의 cox-regression까지 4가지 방법을 C-index를 이용하여 비교하였다.

이 논문에서의 discussion은 다음과 같이 서술하고 있다.

- Faraggi-Simon의 방법은 Design 7, Design 9에 대해서 잘 작동하는 모습을 보였다.
- Liestol et al.의 방법은 Design 7에서 잘 작동하는 모습을 보였다.
- Modified Buckley-James 방법은 Design 5, Design 9에서 잘 작동하는 모습을 보였다. 그러나 Design 4, Design 6, Design 9에서는 성능이 좋지 않은 모습을 보였다. 이 페이퍼에서는 아마도 error function을 잘못 선택했기 때문이라고 생각하고 있었다.
- 모델에 interaction이 포함되지 않았을 때는, 대체로 NN 방법을 이용하는 것이 Cox-regression보다 상대적으로 성능이 더 좋아지는 것을 볼 수 있었다. 이러한 결과는, 생존 분석을 위해 설계된 NN이, 자동으로 interaction을 수용할 수 있는 반면, Cox-regression을 사용하려면 분석가의 통찰력과 경험이 필요함을 나타낸다.
다만 modified Buckley-James 방법은 이에 해당하지 않는다고 한다. 특히 Hidden Layer 수를 늘림으로써 NN 아키텍처를 변경하니, 오히려 성능이 떨어지는 것을 관찰했다고 한다.
- Liestol et al.의 방법은 생존 시간을 얼마나 이산화했는지가, 성능에 크게 영향을 주는 것으로 보인다고 한다. 이산화를 매우 자잘하게 할 수록 성능이 좋아진다고 한다.

20.6.3 의견

생존 분석을 위해 design된 NN 모델이 interaction을 자동으로 수용할 수 있는 장점이 있음을 보여준 연구이다.

Synthetic data를 이용함으로써 discussion에 대한 이야기를 받아들일 수 있다는 것도 장점이다. Real data를 사용하지 않아서 한계가 있다고 이 페이퍼에서는 서술하고 있는데, 이는 Faraggi and Simon과 Liestol et al.의 원래 연구에서 진행되고 있으므로 커버가 된다고 생각한다.

Bibliography

1. 김재균 and 서대출(2009), 생존 분석의 원리와 방법, Neurointervention, Vol. 4, 6-7.
2. 송경일 and 최종수(2007), SPSS 15를 이용한 생존자료의 분석, 한나래출판사.
3. 송인식(2017), 퀄리티 바이블(Quality Bible): 신뢰성 분석 편, 이담.
4. 석진미(2017), 두 범주 결과변수에 대한 예측모형 구축시 연속형 독립변수의 비선형성을 반영하는 방법 비교, 고려대학교 대학원 의학통계학 협동과정, 석사학위논문.
5. 임성빈(????), Survival Analysis via Deep Learning, 내부 자료.
6. 추순규(2015), 생존 자료에서 새로운 인자의 예측력을 평가하는 방법의 비교, 연세대학교 대학원 의학전산통계학 협동과정, 석사학위논문.
7. Antolini L, Boracchi P and Biganzoli E.(2005), A time-dependent discrimination index for survival data, Stat Med., Vol. 24, No. 24, 3927-44.
8. Anny Xiang, Pablo Lapuerta, Alex Ryutov, Jonathan Buckley and Stanley Azen(2000), **Comparison of the performance of neural network methods and Cox regression for censored survival data**, Computational Statistics & Data Analysis, Vol. 34, 243–257.
9. Bain L. J.(1978), Statistical Analysis of Reliability and Life Testing Models, Dekker.
10. Balbir S. Dhillon(1979), A hazard rate model, IEEE Transactions on Reliability, Vol. 28, 180.
11. Balbir S. Dhillon(1981), Life Distributions, IEEE Transactions on Reliability, Vol. 30, No. 5. 457-459.
12. Bain L. J. and Engelhardt M.(1991), **Statistical Analysis of Reliability and Life-Testing Models: Theory**, Marcel Dekker.
13. B. A. Olhausen and D. J. Field(1996), **Emergence of simple-cell receptive field properties by learning a sparse code for natural images**, Nature, Vol. 381, 607-609.
14. Bender, R., T. Augustin and M. Blettner(2005), **Generating survival times to simulate Cox proportional hazards models**, Stat Med, Vol. 24, No. 11, 1713-1723.
15. Benjamin Hofner, Andreas Mayr, Nikolay Robinzonov, Matthias Schmid(????), **Model-based Boosting in R - A Hands-on Tutorial Using the R Package mboost**
16. Birnbaum Z. W. and Saunders S. C.(1968), **A probabilistic interpretation of miner's rule**, SIAM-Jour. Appl. Math., Vol. 16, 637–652.
17. Birnbaum Z. W. and Saunders S. C.(1969), **A new formula of life distribution**, SIAM-Jour. Appl. Prob. Vol. 6, 319–317.
18. K. O. Bowman and L. R. Shenton(1983), **Maximum Likelihood Estimators for the Gamma Distribution Revisited**, Communications in Statistics - Simulation and Computation, Vol. 12.
19. Buckley J. and James I.(1979), **Linear regression with censored data**, Biometrika, Vol. 66, 429-436.
20. Bullinger L., Dohner K., Bair E., Frohling S., Schlenk R. F., Tibshirani R., Dohner H. and Pollack J. R., **Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia**. New England Journal of Medicine, Vol. 350, 1605-1616.
21. Byar D. P. and Green D. K.(1980), **The choice of treatment for cancer patients based on covariates information: application to prostate cancer**, Bulletin of Cancer, Paris, Vol. 67, 477-488.
22. Chambliss L. E. and Diao G.(2006), **Estimation of time-dependent areas under the ROC curve for long-term risk prediction**, Stat Med., Vol. 23, No. 4, 361-387.
23. David Faraggi and Richard Simon(1995), **A Neural Network Model for Survival Data**, Statistics in Medicine, Vol. 14, 73-82.
24. Drzewiecki K. T. and Andersen P. K.(1982), **Survival with malignant melanoma. A regression analysis of prognostic factors**, Cancer, Vol. 49, 2414-2419.
25. Efron B.(1988), **Logistic regression, survival analysis, and the Kaplan—Meier curve**, Journal of the American Statistical Association, Vol. 83, 414—425.
26. Fleming T. and Harrington D.(1991), **Counting Processes and Survival Analysis**, Wiley.

27. Elisa T. Lee and John Wenyu Wang(2013),
Statistical Methods for Survival Data Analysis - 4th Edition, Wiley.
28. Greenwood J. A. and Durand D.(1960),
Aids for fitting the gamma distribution by maximum likelihood, *Technometrics*, Vol. 2, 55-65.
29. M. Gonen and G. Heller(2005),
Concordance probability and discriminatory power in proportional hazards regression, *Biometrika*, Vol. 92.
30. Giuliana Cortese, Thomas H. Scheike and Torben Martinussen(2010),
Flexible survival regression modelling, *Statistical Methods in Medical Research*, Vol. 19, 5–28.
31. Hanley, J. A. and B. J. McNeil(1982),
The meaning and use of the area under a receiver operating characteristic (ROC) curve, *Radiology*, Vol. 143, No. 1, 29-36.
32. Hanley J. A., B. J. McNeil(1983),
A method of comparing the areas under receiver operating characteristic curves derived from the same cases, *Radiology*, Vol. 148, No. 3, 839-843.
33. Heagerty, P. J. and Y. Zheng(2005),
Survival model predictive accuracy and ROC curves, *Biometrics*, Vol. 61, No. 1, 92-105.
34. Hemant Ishwaran, Udaya B. Kogalua, Eugene H. Blackstone and Michael S. Lauer(2008),
Random Survival Forests, *The Annals of Applied Statistics*, Vol. 2, No. 3, 841-860.
35. H. Lee, C. Ekanadham and A. Y. Ng(2008),
Sparse deep belief net model for visual area V2, *Advances in Neural Information Processing Systems*, Vol. 20.
36. H. Lee, R. Grosse, R. Ranganath and A. Y. Ng(2009),
Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, *Proceedings of the 26th Annual International Conference on Machine Learning*, 609-616.
37. Hothorn T., Buhlmann P., Dudoit S., Molinaro A. and van der Laan M. J.(2006).
Survival ensembles, *Biostatistics*, Vol. 7, No. 3, 355-373.
38. Ishwaran H., Blackstone E. H., Pothier C. and Lauer M. S.(2004),
Relative risk forests for exercise heart rate recovery as a predictor of mortality, *J. Amer. Statist. Assoc.*, Vol. 99, 591–600.
39. Ishwaran H. and Kogalur U. B.(2007), **Random survival forests for R**, *Rnews*, Vol. 7, No. 2, 25-31.
40. Jared L. Katzman, Uri Shaham, Jonathan Bates, Alexander Cloninger, Tingting Jiang and Yuval Kluger(2016),
Deep Survival: A Deep Cox Proportional Hazards Networks, *arXiv:1606.00931v2 [stat.ML]* 25 Oct 2016.
41. Kattan M.(2003),
Comparison of Cox regression with other methods for determining prediction models and nomograms, *J. Urol.*, Vol. 170, 6–10.
42. Kalbfleisch J. D. and Prentice R. L.(1980), **The Statistical Analysis of Failure Time Data**, Wiley.
43. Knut Liestol, Per Kragh Andersen and Ulrich Andersen(1994),
Survival analysis and neural nets, *Statistics in medicine*, Vol. 13, No. 12, 1189–1200.
44. Koziol J. A., Jia Z.(2009), **The concordance index C and the Mann-Whitney parameter $Pr(X > Y)$ with randomly censored data.**, *Biometrical Journal*, Vol. 51, No. 3, 467-474.
45. M. Ito and H. Komatsu(2004),
Representation of angles embedded within contour stimuli in area V2 of macaque monkeys, *The Journal of Neuroscience*, Vol. 24, No. 13, 3313-3324.
46. Military Specification (MIL)-HDBK-338A, **Electronic Reliability Design Handbook** (DOD, 12 October 1988).
47. Margaux Luck, Tristan Sylvain, Meloise Cardinal, Andrea Lodi and Yoshua Bengio(2017),
Deep Learning for Patient-Specific Kidney Graft Survival Analysis, *arXiv:1705.10245v1 [cs.LG]* 29 May 2017.
48. Odd O. Aalen, Per Kragh Andersen, Ørnulf Borgan, Richard D. Gill, Niels Keiding (2010),
History of applications of martingales in survival analysis, *Electronic Journal for History of Probability and Statistics*, Vol. 5, No. 1. 28.
49. Pencina, M. J., R. B. D'Agostino, Sr., R. B. D'Agostino, Jr. and R. S. Vasan(2008).
Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond, *Stat Med*, Vol. 27, No. 2, 157–72.
50. Pencina, M. J., R. B. D'Agostino, Sr. and E. W. Steyerberg(2011).
Extensions of net reclassification improvement calculations to measure usefulness of new biomarkers, *Stat Med*, Vol. 30, No. 1, 11–21.

51. Peter M. Ravdin and Gary M. Clark(1992).
A practical application of neural network analysis for predicting outcome of individual breast cancer patients, Breast Cancer Research and Treatment, Vol. 22, No. 3, 285–293.
52. Ping Wang, Yan Li and Chandan K. Reddy(2017),
Machine Learning for Survival Analysis: A Survey, ACM Computing Surveys, Vol. 1, No. 1.
53. Prentice R. L. and Gloeckler L. A.(1978),
Regression analysis of grouped survival data with application to breast cancer data, Biometrics, Vol. 34, 57-67.
54. Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean and A. Y. Ng(2012),
Building high-level features using large scale unsupervised learning, Proceedings of the 29th International Conference on International Conference on Machine Learning, 507-514.
55. Roger Newson(2007), **Parameters behind "nonparametric" statistics: Kendall's tau, Somers' D and median differences**, The Stata Journal, Vol. 2, No. 1, 45-64.
56. Rossi P. H., Berk R. A. and Lenihan K. J.(1980),
Money, Work and Crime: Some Experimental Results, Academic Press.
57. Rupert Miller and Jerry Halpern(1982), **Regression with Censored Data**, Biometrika, Vol. 69, No. 3, 521-31.
58. Saleh J. H. and Marais Ken(2006), **Highlights from the Early (and pre-) History of Reliability Engineering**, Reliability Engineering and System Safety, Vol. 91, No. 2, 249-256.
59. Salvia A. A.(1985), **Reliability application of the alpha distribution**, IEEE Transactions on Reliability, Vol. 34, 251-252.
60. Stacy E. W.(1962), **A generalization of the gamma distribution**, Ann. Math. Statist., Vol. 33, 1187–1192.
61. Steyerberg E. W.(2009), **Clinical Prediction Models A Practical Approach to Development, Validation, and Updating**, Springer.
62. Terry Therneau, Cynthia Crowson and Elizabeth Atkinson(2014), **Using Time Dependent Covariates and Time Dependent Coefficients in the Cox Model**
63. Weibull W. A.(1939), **A Statistical Theory of the Strength of Materials**, Ingenjörs Ventenskaps Akademien Ilandlänger, Vol. 151, 5-45.
64. Van Der Laan M. J. and Dudoit, S.(2003),
Unified cross-validation methodology for selection among estimators: finite sample results, asymptotic optimality, and applications,
Technical Report 130, Division of Biostatistics, University of California, Berkeley, California,
<http://www.bepress.com/ucbbiostat/paper130>.
65. Van Der Laan M. J. and Robins J. M.(2003),
Unified Methods for Censored Longitudinal Data and Causality, Springer.
66. Venable W. N. and Ripley B. D.(2004), **Modern Applied Statistics with S-Plus, 4th edition**, Springer-Verlag.