Survival Analysis and TCGA data

Helian Feng

April 16

Survival Analysis

- In some studies, the response variable of interest is the length of time between an initial observation and the occurrence of a subsequent event
- The event is often called a *failure*
- The time from the initial observation until failure is called the *survival time*
- Examples: Time from birth until death, time from start of treatment until serious adverse event, time from randomization to relapse or death, time from entry in a cohort study until myocardial infarction

Goals of Survival Analysis

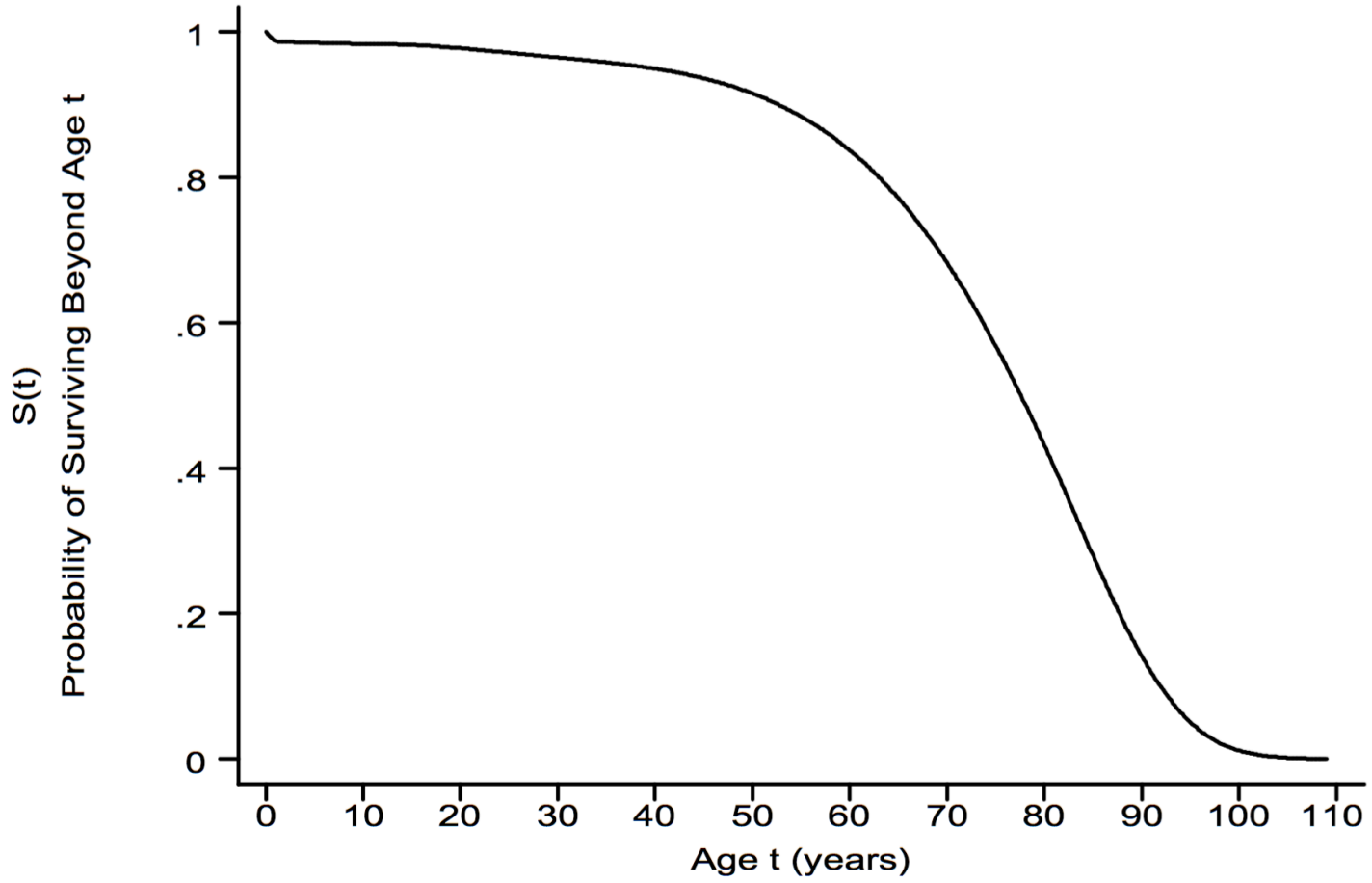
- To estimate the distribution of survival times for a population
- To test the equality of survival distributions (e.g., treated vs. control group, smokers vs. nonsmokers)
- To estimate and control for the effects of other covariates when investigating the relationship between a predictor variable and survival time

Survival Function

Let T_i be the time to event for patient i. There exist three interrelated functions which categorize the distribution of T_i:

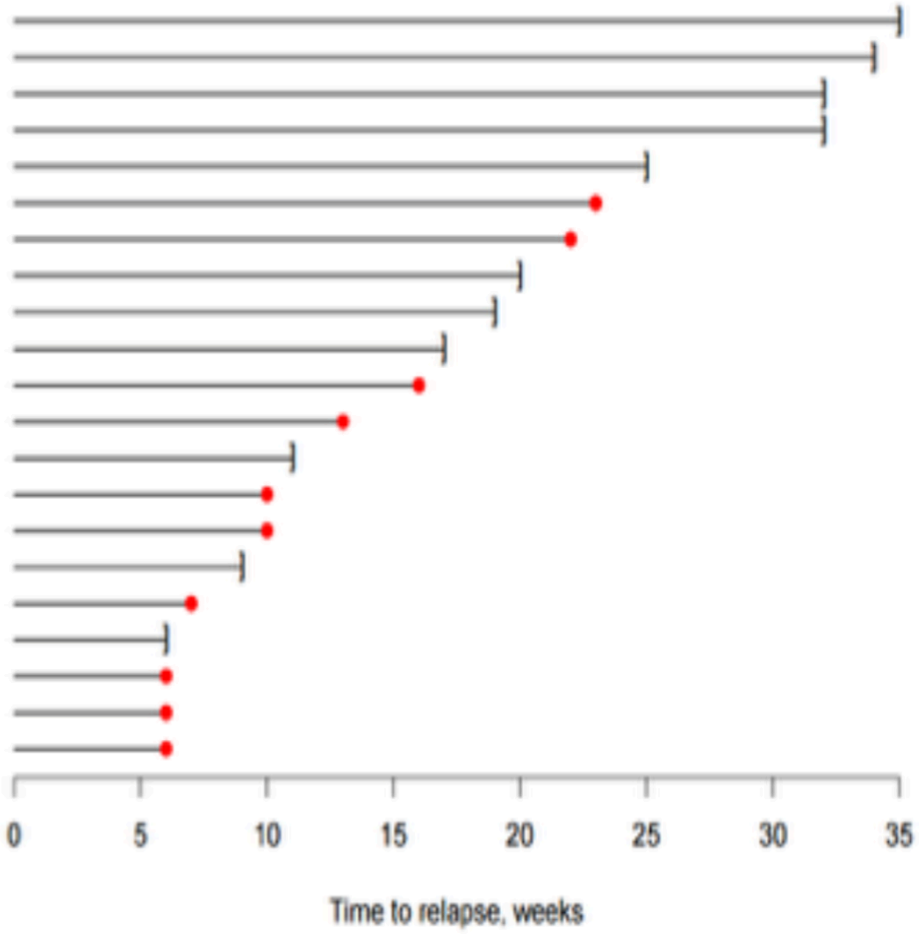
1. **Probability density function:** $f(t) = P(T_i = t) = \lambda(t)S(t)$
2. **Survival function:** $S(t) = P(T_i > t) = 1 - P(T_i \leq t) = 1 - F(t)$ where $F(t)$ describes the CDF of our previously defined probability density function.
3. **Hazard function:** $\lambda(t) = \frac{f(t)}{S(t)} = -\frac{d}{dt}log(S(t))$. This can be thought of as the instantaneous probability of experiencing an event at time t, given that one has not experienced the event prior to time t.

Survival Curve



Censored Data

- Since most studies occur over a finite time period, the event of interest may not have occurred for some subjects during the study period
- All that is known is that the time to an event *T* is greater than the period of follow-up *C*, where *C* is called the *censoring time*
- For subjects who have an event during the study period, we have the actual event time *T*
- **Right Censoring**
  - We know that the event time *T* is greater than the censoring time *C*
  - This is the most common form of censoring



Graphing survival curves

- The Kaplan-Meier estimator is a non-parametric estimator for survival curves. Using the K-M estimator, we do not have to make any distributional assumptions about survival times. This method is intuitive, but has limitations in that we cannot account for covariate data.
- The Kaplan-Meier estimate calculates empirical probabilities at every follow-up time using the formula
- $$P(T_i > t_j) = P(T_i > t_j | T_i > t_{j-1}) P(T_i > t_{j-1})$$
- where j is the index of follow-up times. Naturally, by expanding the formula several times, we arrive at
- $$P(T_i > t_j) = \prod_{k=1}^{j} P(T_i > t_k | T_i > t_{k-1}) P(T_i > t_{k-1})$$
- by our assumptions at the beginning of the study.
- Let's do an example below:
- Let the survival times of 20 patients be: 1, 1+, 2, 3, 4, 4, 4, 5, 5+, 5+, 6, 6, 6+, 7, 7, 7, 8, 8+, 8+, 9. Where '+' indicates censoring (if an observation is designated '1+' it means that they had their event after time period 1). Fill out the following table and graph this curve by hand.

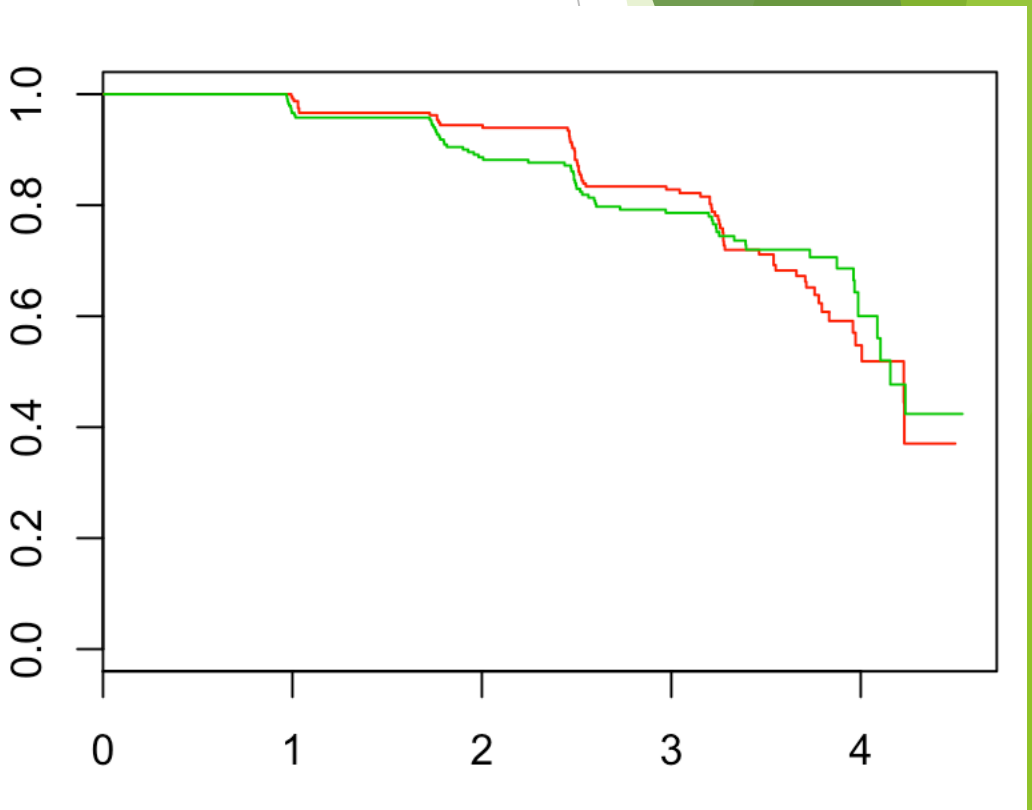
Kaplan-Meier

Table 1: My caption

	Risk Set	Events	Conditional Survival	Unconditional Survival
1	20	1	19/20	0.95
2	18	1	17/18	0.897
3	17	1	16/17	0.844
4	16	3	13/16	0.686
5	13	1	12/13	0.633
6	10	2	8/10	0.507
7	7	3	4/7	0.290
8	4	1	3/4	0.217
9	1	1	0	0

R demonstration

- Variables on the file srt.dat are: id (a subject id), sorb (randomized treatment assignment, 1=sorbinil, 0=placebo), tgh (total glycosylated hemoglobin in percent), dur (duration of diabetes in years since diagnosis), sex (2=female, 1=male), fup (duration of follow-up in years until progression of diabetic retinopathy or end of follow-up), and status (1=diabetic retinopathy progressed, 0=no progression)



Testing differences

- The log-rank test is a non-parametric test of the difference in survival curves.

1. Create K 2x2 tables, where K is the total number of follow-up times.
2. At each time point t_k, note the number of events (d_{jk}) and patients in the risk set (n_{jk}) for each group $j = 1, 2$. Let d_k be the total number of events at t_k and n_k be similarly the total number of people in the risk set.
3. Calculate $O = \sum_k d_{1k}$, $E = \sum_k \frac{n_{1k} d_k}{n_k}$ and $V = \sum_k \frac{n_{1k} n_{2k} d_k (n_k - d_k)}{n_k^2}$.
4. Compare $\frac{(O-E)^2}{V}$ to a chi-square distribution with degrees of freedom $= 1$.

	Failure		
Group	Yes	No	Total
Maintenance	d_{1i}	$n_{1i} - d_{1i}$	n_{1i}
Control	d_{2i}	$n_{2i} - d_{2i}$	n_{2i}
Total	d_i	$n_i - d_i$	n_i

- The restrictions of this test are that V must be larger than 5 and the proportional hazards assumption holds (aka the survival curves of the two groups do not overlap). If this assumption is violated, the test is underpowered.

R demonstration of log-rank test

Cox regression model

- The Cox regression model is a semi-parametric regression model for survival data. It is semi-parametric in that it makes parametric assumptions about a linear combination of covariates, but the baseline hazard is not required to be a known distribution:

$$h_i(t) = h_0(t)\exp(\beta\ x_i)$$

$h(t|X=1) = h_0(t)\exp(\beta)$ if a patient is in the maintained group,

$h(t|X=0) = h_0(t)$ if a subject is in the control group.

$$\frac{h(t|X=1)}{h(t|X=0)} = \exp(\beta) = \text{hazard ratio},$$

or $\beta = \log(\text{hazard ratio})$.

TCGA: The Cancer Genome Atlas

- http://cancergenome.nih.gov/
- Data portal:
- https://portal.gdc.cancer.gov/
-

Microarray analysis review

- require(affy)
- require(affyPLM)
- require(limma)
- fit<-lmFit(GBM_expr,design)
- fit<-eBayes(fit)
- top=topTable(fit,lfc = p.value =,number =)

Kmeans review

- kmean_all<-kmeans(t(as.matrix(dat)),3)
- Visualize with top two PCs
- pc.cr <- prcomp(t(dat))
- pca1 <-pc.cr$x[,1]
- pca2 <-pc.cr$x[,2]

Epigenetics

- Logit transformation
- Then use lmFit and eBayes as microarray data

Mutation analysis

1. Summarize the mutations in each sample
2. Group by subtypes
3. Find the different ones

Python programming tip

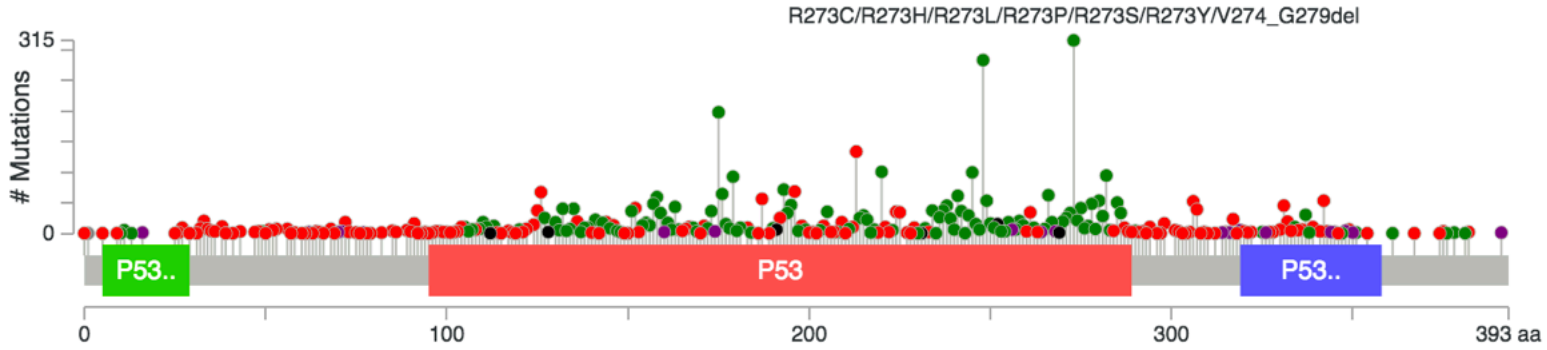
- Exporting subtype list from R
- Create mutation count dict for all; subtype1; subtype2
- Loop through file summarize number of appearance for each mutation
- Sort the results according to number of appearance and output the file

```
for filename in os.listdir(directory):
    #print '2'
    #print ( filename)
    for strLine in open(os.path.join(directory, filename) ):
        astrLine = strLine.strip( ).split( "\t" )
        if astrLine[0] not in mutationCount.keys():
            mutationCount[astrLine[0]] = 1
        else:
            mutationCount[astrLine[0]] += 1
        if subtype[filename] == subtype1:
            if astrLine[0] not in mutationCount_subtype1.keys():
                mutationCount_subtype1[astrLine[0]] = 1
            else:
                mutationCount_subtype1[astrLine[0]] += 1
        else:
            if astrLine[0] not in mutationCount_subtype2.keys():
                mutationCount_subtype2[astrLine[0]] = 1
            else:
                mutationCount_subtype2[astrLine[0]] += 1
```

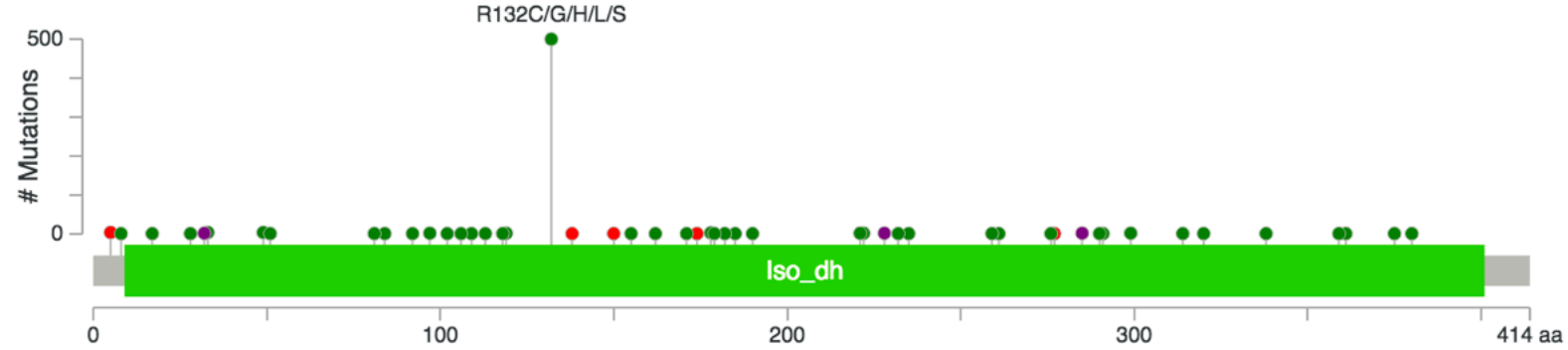
```
def writeTop30MutationCount(filename,mutationCount):
    fout=open(filename,"w")
    i = 0
    for w in sorted(mutationCount, key=mutationCount.get, reverse=True):
      if i < 30:
        newline = [w, str(mutationCount[w] )]
        fout.write("\t".join( newline))
        fout.write("\n")
        i +=  1
    fout.close()

writeTop30MutationCount("MutationCount_top30.txt",mutationCount)
```

Gain/Loss function mutation

- Loss of function mutation



- Gain of function mutation



Thanks