Lecture 3 Code

Alex Hoagland

2025-01-27

# I like to include several additional notes in the header of my files here:  
#  
# Last modified: 2024-01-23  
#  
### PURPOSE:  
 # Lecture 3 code and output file  
#  
### NOTES:  
 # - uses the Tidyverse package and Dplyr  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.4.2

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(NHANES)

## Warning: package 'NHANES' was built under R version 4.4.2

### Covariance and Correlation

mydata <- NHANES  
cov(mydata$DaysMentHlthBad, mydata$Age,use="pairwise.complete.obs") # Base covariance -- but in units of what?

## [1] -4.848859

cov(mydata$DaysMentHlthBad/7, mydata$Age,use="pairwise.complete.obs")

## [1] -0.6926941

cor(mydata$DaysMentHlthBad, mydata$Age,use="pairwise.complete.obs") # This makes more sense!

## [1] -0.03200316

cor(mydata$DaysMentHlthBad/7, mydata$Age,use="pairwise.complete.obs") # This makes more sense!

## [1] -0.03200316

# What makes this number useful for you? What would you think about changing to make this number more useful, given our results above?

# Random data

When working with a sample of data, we want to understand something about the *randomness* of that data. Hence, we talk about sampling distributions.

## Example: Sampling from a Bernoulli distribution

Just as a simple example, suppose that we have data with two possible outcomes: poor mental health day or not. How can we express this distribution simply?

# First, sample some data   
sampledata <- rbinom(n=1000,size=1,prob=0.1)  
hist(sampledata) # What does this tell us? What should I change above to change the shape of the graph?

# Second, how can we go from the data we have to an indicator of the probability of poor mental health p?  
hist(mydata$DaysMentHlthBad)

mydata %>% mutate(anybad = ifelse(DaysMentHlthBad > 0 & !is.na(DaysMentHlthBad),   
 1, 0)) %>%   
 ggplot(aes(x=anybad)) +   
 geom\_histogram(aes(y = (..count..)/sum(..count..))) # Does this look like what we had above?

## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.  
## ℹ Please use `after\_stat(count)` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

So if we had to use this data, what would our estimate of be?

## Now what about Binomial?

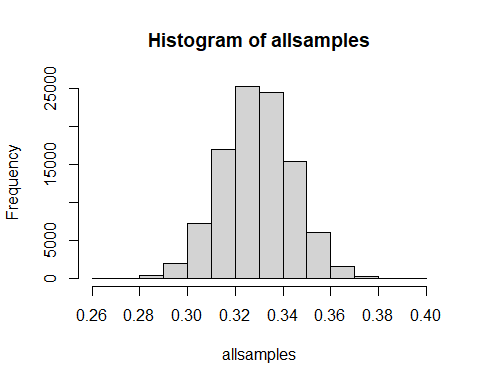
We simplified our data a bit above. But what if we wanted to know about the number of poor mental health days in a month (intensive margin) rather than just the probability of poor mental health at all (extensive margin)?

# Let's calculate the empirical average successes/fails per person  
avgsuccess <- mean(mydata$DaysMentHlthBad, na.rm=T)  
avgfails <- mean(30-mydata$DaysMentHlthBad, na.rm=T)  
p <- avgsuccess/30 # This is the average # of poor mental health days in a month  
  
hist(mydata$DaysMentHlthBad) # Can we match this to a binomial distribution?

sampledata <- rbinom(n=100000,size=30,prob=p)  
hist(sampledata) # What does this tell us? Does our data follow a binomial distribution?

How could we test this empirically? Let’s sample the predicted probabilities on just the extensive margin and see how they converge over multiple random samples:

# Let's sample, with replacement, from our dataset.   
# Each time we sample, we will compute p = the average Pr(Any Poor Mental Health)  
# Then we will do this 100,000 times  
# And look at the distribution of estimated p's.   
  
mydata <- mydata %>% mutate(anybad = ifelse(DaysMentHlthBad > 0 & !is.na(DaysMentHlthBad),   
 1, 0))  
allsamples <- rep(NA, 1e5) # empty frame to store results  
for (i in 1:100000) {  
 test <- sample(mydata$anybad, replace=TRUE, size=1000) # 10% random sample  
 allsamples[i] <- mean(test)  
 rm(test)  
}  
hist(allsamples) # What distribution do we have here? Why?



# This gives us two things: the estimated average value and the \*uncertainty\* around that value  
mean(allsamples)

## [1] 0.3297445

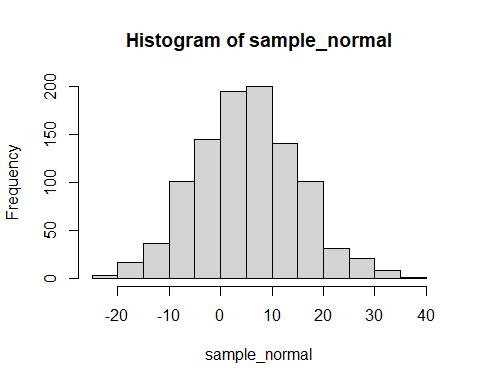
sd(allsamples)

## [1] 0.01481942

## Calculating values from the normal distribution

Note: this is all very similar across the “main” distributions that R knows. We just use the normal distribution here as an example, but you can use Uniform, Bernoulli, Binomial, Exponential, Gamma, or any other distribution in basically the same way!

# If you want to sample from a normal distribution:  
sample\_normal <- rnorm(n=1000,mean=5,sd=10)  
hist(sample\_normal) # think: how would you standardize this? Can we add code here to do that?



# Standardize sample\_normal  
sample\_normal2 <- (sample\_normal - mean(sample\_normal))/sd(sample\_normal)  
sample\_normal3 <- (sample\_normal-5)/10  
  
# First, pdf  
pnorm(q=0,mean=0,sd=1)

## [1] 0.5

pnorm(q=10,mean=5,sd=10) # what is the probability that a normal draw is less than 10?

## [1] 0.6914625

# Second, cdf  
dnorm(x=10,mean=5,sd=10) # this gives the density

## [1] 0.03520653

# Third, quantiles  
  
# Identify the 5th and 95th percentiles of a normal distribution with mean 5 and sd 10  
qnorm(p=c(0.05,0.95),mean=5,sd=10)

## [1] -11.44854 21.44854

qnorm(p=c(.1,.2,.3,.4,.5,.6,.7,.8,.9,1),mean=5,sd=10) # quantiles of the distribution

## [1] -7.8155157 -3.4162123 -0.2440051 2.4665290 5.0000000 7.5334710  
## [7] 10.2440051 13.4162123 17.8155157 Inf

# Fourth, critical values  
qnorm(p=c(.025,.975),mean=5,sd=10) # two-sided quantiles of the distribution

## [1] -14.59964 24.59964

## Sampling and standard errors

Let’s suppose that we have a population (we don’t) from which we repeatedly sample and estimate a mean. For this one, we will use data on COVID prevalence.

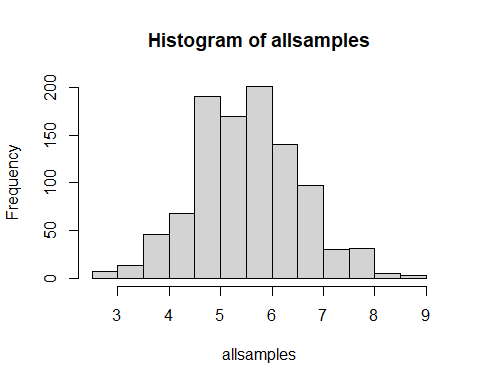
# install.packages("medicaldata")  
library(medicaldata)

## Warning: package 'medicaldata' was built under R version 4.4.2

data(package = "medicaldata") # this will show you the full list of data sets  
# https://higgi13425.github.io/medicaldata/  
  
# Let's use the covid-testing data  
mydata <- covid\_testing # Suppose that this is the population  
  
# If we want to take samples of 500 patients each  
sample1 <- mydata %>% sample\_n(500, replace=TRUE)  
sample1 %>% mutate(positive = (result == "positive")\*100) %>% # generate new variable (in %)  
 ungroup() %>% select(positive) %>% summarize\_all(mean) # summarize average value

## # A tibble: 1 × 1  
## positive  
## <dbl>  
## 1 4.8

# What if we take 100 of these samples?  
allsamples <- rep(NA, 1000) # empty frame to store results  
for (i in 1:1000) {  
 test <- mydata %>% sample\_n(500, replace=TRUE)  
 allsamples[i] <- test %>% mutate(positive = (result == "positive")\*100) %>% # generate new variable (in %)  
 ungroup() %>% select(positive) %>% summarize\_all(mean) %>% as.numeric() # summarize average value  
 rm(test)  
}  
hist(allsamples)



sd(allsamples) # This is a measure of variability (how would you think about putting it on the plot?)

## [1] 1.007475

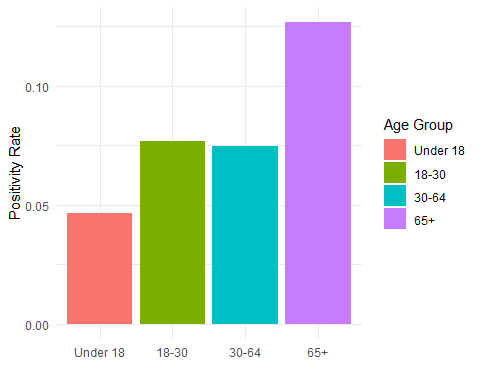
This isn’t that different from the days of poor mental health code above. But what if we wanted to compute standard errors more directly?

# What about calculating SEs directly in the pouplation?  
mydata <- mydata %>% mutate(positive = (result == "positive"))  
se1 <- sd(mydata$positive)/sqrt(nrow(mydata)) # this is the standard error for our mean  
  
# Note: if we actually knew something about the sampling distribution, we could refine these estimates.   
# For examlpe, suppose we know these tests are Bernoulli with p= the average positivity rate (0.06). Then, we can update our SE:   
se2 <- sqrt(0.06\*(1-0.06)/nrow(mydata)) # this is the standard error for our mean. Does it differ much?

Now, let’s talk about *plotting* uncertainty. What if we wanted to plot average positive rate across age bins?

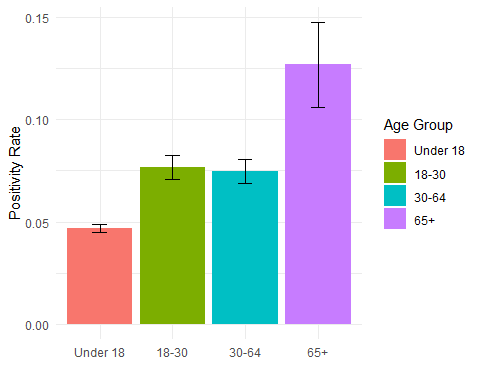
# first, create some age bins  
mydata <- mydata %>% mutate(agebin = ifelse(age < 18, 1,  
 ifelse(age >= 18 & age < 30, 2,  
 ifelse(age >= 30 & age < 65, 3, 4))))  
  
# now without uncertainty, how can we plot?  
mydata %>% select(agebin, positive) %>%  
 group\_by(agebin) %>% summarize(positive= mean(positive)) %>%   
 mutate(agebin = factor(agebin, levels=1:4,labels=c("Under 18", "18-30", "30-64", "65+"))) %>% # make sure this is a factor  
 ggplot(aes(x=agebin,y=positive,fill=agebin)) +   
 geom\_bar(position = "dodge", stat = "summary") +   
 labs(fill="Age Group", x="", y="Positivity Rate") + theme\_minimal()

## No summary function supplied, defaulting to `mean\_se()`



# now how to add uncertainty -- error bars  
mydata %>% select(agebin, positive) %>%  
 group\_by(agebin) %>% summarize(mean= mean(positive), sd=sd(positive, na.rm=T), n=n()) %>%  
 mutate(se = sd/sqrt(n)) %>% # calculate the standard error  
 mutate(agebin = factor(agebin, levels=1:4,labels=c("Under 18", "18-30", "30-64", "65+"))) %>% # make sure this is a factor  
 ggplot(aes(x=agebin,y=mean,fill=agebin)) +   
 geom\_bar(position = "dodge", stat = "summary") +   
 geom\_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.2,position=position\_dodge(.9)) +   
 labs(fill="Age Group", x="", y="Positivity Rate") + theme\_minimal()

## No summary function supplied, defaulting to `mean\_se()`



The main question, though, is: how much uncertainty should I show? And why?

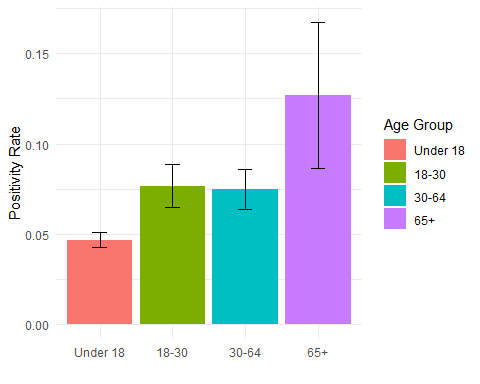
## Calculating confidence intervals

Note that we already built a confidence interval! All we need to do is appropriately adjust the standard error in the graph above. So how do we interpret it?

Suppose we want to calculate these directly. Can we? Yes!

# First, tweak our graph  
mydata %>% select(agebin, positive) %>%  
 group\_by(agebin) %>% summarize(mean= mean(positive), sd=sd(positive, na.rm=T), n=n()) %>%  
 mutate(se = sd/sqrt(n)) %>% # calculate the standard error  
 mutate(agebin = factor(agebin, levels=1:4,labels=c("Under 18", "18-30", "30-64", "65+"))) %>% # make sure this is a factor  
 ggplot(aes(x=agebin,y=mean,fill=agebin)) +   
 geom\_bar(position = "dodge", stat = "summary") +   
 geom\_errorbar(aes(ymin=mean-1.96\*se, ymax=mean+1.96\*se), width=.2,position=position\_dodge(.9)) +   
 labs(fill="Age Group", x="", y="Positivity Rate") + theme\_minimal()

## No summary function supplied, defaulting to `mean\_se()`



# Does anything change in our interpretation here?  
  
# What's the overall confidence interval?   
prop.test(sum(mydata$positive),nrow(mydata)) # This is the easiest way to do this for binary variables

##   
## 1-sample proportions test with continuity correction  
##   
## data: sum(mydata$positive) out of nrow(mydata), null probability 0.5  
## X-squared = 12255, df = 1, p-value < 2.2e-16  
## alternative hypothesis: true p is not equal to 0.5  
## 95 percent confidence interval:  
## 0.05218934 0.05947280  
## sample estimates:  
## p   
## 0.05572018

prop.test(sum(mydata$positive),nrow(mydata), conf.level = 0.9)

##   
## 1-sample proportions test with continuity correction  
##   
## data: sum(mydata$positive) out of nrow(mydata), null probability 0.5  
## X-squared = 12255, df = 1, p-value < 2.2e-16  
## alternative hypothesis: true p is not equal to 0.5  
## 90 percent confidence interval:  
## 0.05273730 0.05885952  
## sample estimates:  
## p   
## 0.05572018

prop.test(sum(mydata$positive),nrow(mydata), conf.level = 0.99)

##   
## 1-sample proportions test with continuity correction  
##   
## data: sum(mydata$positive) out of nrow(mydata), null probability 0.5  
## X-squared = 12255, df = 1, p-value < 2.2e-16  
## alternative hypothesis: true p is not equal to 0.5  
## 99 percent confidence interval:  
## 0.05113417 0.06068836  
## sample estimates:  
## p   
## 0.05572018

# confidence intervals across agebins  
prop.test(sum(mydata[which(mydata$agebin==1),]$positive),nrow(mydata[which(mydata$agebin==1),]))

##   
## 1-sample proportions test with continuity correction  
##   
## data: sum(mydata[which(mydata$agebin == 1), ]$positive) out of nrow(mydata[which(mydata$agebin == 1), ]), null probability 0.5  
## X-squared = 9184.1, df = 1, p-value < 2.2e-16  
## alternative hypothesis: true p is not equal to 0.5  
## 95 percent confidence interval:  
## 0.04297599 0.05089884  
## sample estimates:  
## p   
## 0.04677996

prop.test(sum(mydata[which(mydata$agebin==2),]$positive),nrow(mydata[which(mydata$agebin==2),]))

##   
## 1-sample proportions test with continuity correction  
##   
## data: sum(mydata[which(mydata$agebin == 2), ]$positive) out of nrow(mydata[which(mydata$agebin == 2), ]), null probability 0.5  
## X-squared = 1380.7, df = 1, p-value < 2.2e-16  
## alternative hypothesis: true p is not equal to 0.5  
## 95 percent confidence interval:  
## 0.06542943 0.08973704  
## sample estimates:  
## p   
## 0.07672369

prop.test(sum(mydata[which(mydata$agebin==3),]$positive),nrow(mydata[which(mydata$agebin==3),]))

##   
## 1-sample proportions test with continuity correction  
##   
## data: sum(mydata[which(mydata$agebin == 3), ]$positive) out of nrow(mydata[which(mydata$agebin == 3), ]), null probability 0.5  
## X-squared = 1557.4, df = 1, p-value < 2.2e-16  
## alternative hypothesis: true p is not equal to 0.5  
## 95 percent confidence interval:  
## 0.06413425 0.08683076  
## sample estimates:  
## p   
## 0.07470998

prop.test(sum(mydata[which(mydata$agebin==4),]$positive),nrow(mydata[which(mydata$agebin==4),]))

##   
## 1-sample proportions test with continuity correction  
##   
## data: sum(mydata[which(mydata$agebin == 4), ]$positive) out of nrow(mydata[which(mydata$agebin == 4), ]), null probability 0.5  
## X-squared = 143.27, df = 1, p-value < 2.2e-16  
## alternative hypothesis: true p is not equal to 0.5  
## 95 percent confidence interval:  
## 0.09018236 0.17503554  
## sample estimates:  
## p   
## 0.1269231

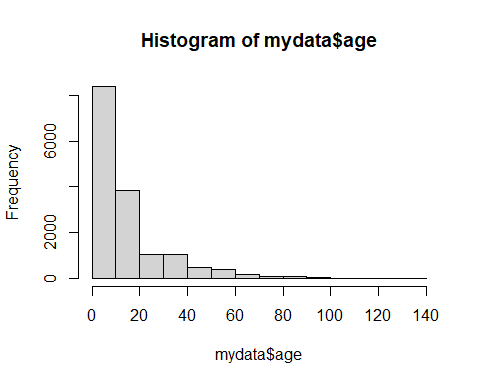
### Confidence intervals for proportions vs. means

Let’s get a little more specific into confidence intervals for proportions vs. means. Above, we really used a proportion. What if we wanted to use a truly continuous variable?

# Let's look at CIs for age  
t.test(mydata$age) # This gives us extraneous information but we'll get there!

##   
## One Sample t-test  
##   
## data: mydata$age  
## t = 107.33, df = 15523, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 13.93033 14.44862  
## sample estimates:  
## mean of x   
## 14.18947

hist(mydata$age) # do we want the CI for the mean here?



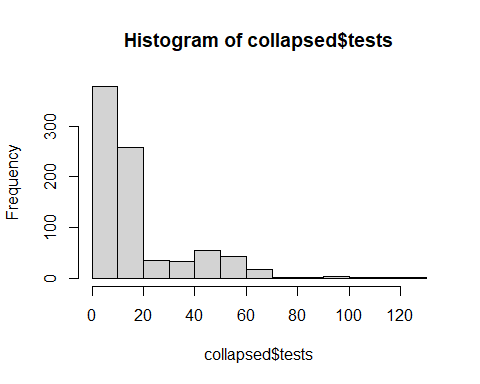
wilcox.test(mydata$age, conf.int = TRUE) # This gives us a CI for the median

##   
## Wilcoxon signed rank test with continuity correction  
##   
## data: mydata$age  
## V = 116456691, p-value < 2.2e-16  
## alternative hypothesis: true location is not equal to 0  
## 95 percent confidence interval:  
## 10.99995 11.49997  
## sample estimates:  
## (pseudo)median   
## 11.00002

### CIs for Poisson data

Suppose that we wanted to say something about *how many* tests each individual got, rather than the outcome of tests. That is, what if we were worried about differential access to testing?

# to get some fake data on this, let's collapse by first name in the dataset  
collapsed <- mydata %>% group\_by(fake\_first\_name) %>% summarize(tests = n())  
hist(collapsed$tests) # Look Poisson enough?



summary(collapsed$tests) # note the skew

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.00 8.00 11.00 18.66 19.00 121.00

# but we don't need a package to get SEs!  
collapsed %>% ungroup() %>% summarize(mean = mean(tests), n = n()) %>%  
 mutate(se = sqrt(mean/n)) %>% select(se)

## # A tibble: 1 × 1  
## se  
## <dbl>  
## 1 0.150

# Now how can you plot this? Make a CI? Try it!

### CIs for differences in means

Finally, let’s test for the difference between means/proportions across two groups: men and women. We’ll look at a continuous variable (time to first test) and a binary variable (positivity rates)

# First, let's do a difference in means  
t.test(pan\_day ~ gender, data = mydata, var.equal = FALSE)

##   
## Welch Two Sample t-test  
##   
## data: pan\_day by gender  
## t = 0.3889, df = 15520, p-value = 0.6974  
## alternative hypothesis: true difference in means between group female and group male is not equal to 0  
## 95 percent confidence interval:  
## -0.6865042 1.0263409  
## sample estimates:  
## mean in group female mean in group male   
## 63.29277 63.12285

# Second, let's do a difference in proportions  
# install.packages('tidymodels') # note: this takes a while  
library(tidymodels)

## Warning: package 'tidymodels' was built under R version 4.4.2

## ── Attaching packages ────────────────────────────────────── tidymodels 1.2.0 ──

## ✔ broom 1.0.7 ✔ rsample 1.2.1  
## ✔ dials 1.3.0 ✔ tune 1.2.1  
## ✔ infer 1.0.7 ✔ workflows 1.1.4  
## ✔ modeldata 1.4.0 ✔ workflowsets 1.1.0  
## ✔ parsnip 1.2.1 ✔ yardstick 1.3.1  
## ✔ recipes 1.1.0

## Warning: package 'dials' was built under R version 4.4.2

## Warning: package 'infer' was built under R version 4.4.2

## Warning: package 'modeldata' was built under R version 4.4.2

## Warning: package 'parsnip' was built under R version 4.4.2

## Warning: package 'recipes' was built under R version 4.4.2

## Warning: package 'rsample' was built under R version 4.4.2

## Warning: package 'tune' was built under R version 4.4.2

## Warning: package 'workflows' was built under R version 4.4.2

## Warning: package 'workflowsets' was built under R version 4.4.2

## Warning: package 'yardstick' was built under R version 4.4.2

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ scales::discard() masks purrr::discard()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ recipes::fixed() masks stringr::fixed()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ yardstick::spec() masks readr::spec()  
## ✖ recipes::step() masks stats::step()  
## • Search for functions across packages at https://www.tidymodels.org/find/

testdata <- mydata %>% ungroup() %>% select(positive, gender) %>%   
 mutate(positive = as.factor(positive),   
 gender = as.factor(gender))  
# prop\_test(testdata,detailed=TRUE)

## Survival Curves

Let’s plot how long people “last” before a test across groups (let’s go back to agebins). We can use the “pan\_day” variable here since it already has a helpful time 0.

# Need some packages  
# install.packages(c("survival", "survminer"))  
library("survival")  
library("survminer")

## Warning: package 'survminer' was built under R version 4.4.2

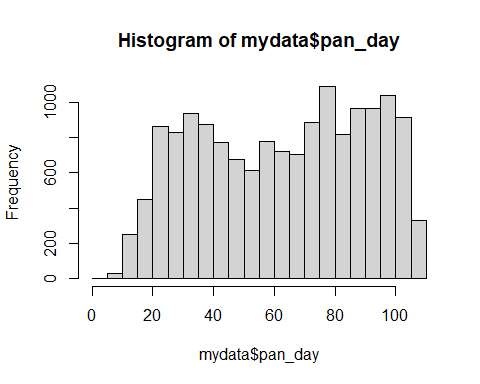
## Loading required package: ggpubr

## Warning: package 'ggpubr' was built under R version 4.4.2

##   
## Attaching package: 'survminer'

## The following object is masked from 'package:survival':  
##   
## myeloma

# Need to define a "time 0"  
hist(mydata$pan\_day)



# Now we can do survival based on bins  
fit <- survfit(Surv(pan\_day) ~ agebin, data = mydata)  
print(fit) # What do we take from this?

## Call: survfit(formula = Surv(pan\_day) ~ agebin, data = mydata)  
##   
## n events median 0.95LCL 0.95UCL  
## agebin=1 11180 11180 65.0 64 66  
## agebin=2 1929 1929 42.0 39 44  
## agebin=3 2155 2155 75.0 73 77  
## agebin=4 260 260 91.5 89 94

# Let's plot!  
ggsurvplot(fit,  
 pval = TRUE, conf.int = TRUE,  
 risk.table = TRUE, # Add risk table  
 risk.table.col = "strata", # Change risk table color by groups  
 linetype = "strata", # Change line type by groups  
 surv.median.line = "hv", # Specify median survival  
 ggtheme = theme\_bw(), # Change ggplot2 theme  
 palette = c("#E7B800", "#2E9FDF", "palegreen", "mediumpurple1"))

## Warning in geom\_segment(aes(x = 0, y = max(y2), xend = max(x1), yend = max(y2)), : All aesthetics have length 1, but the data has 4 rows.  
## ℹ Please consider using `annotate()` or provide this layer with data containing  
## a single row.

## Warning in geom\_segment(aes(x = 0, y = max(y2), xend = max(x1), yend = max(y2)), : All aesthetics have length 1, but the data has 4 rows.  
## ℹ Please consider using `annotate()` or provide this layer with data containing  
## a single row.  
## All aesthetics have length 1, but the data has 4 rows.  
## ℹ Please consider using `annotate()` or provide this layer with data containing  
## a single row.  
## All aesthetics have length 1, but the data has 4 rows.  
## ℹ Please consider using `annotate()` or provide this layer with data containing  
## a single row.

