Computer Lab 9: Counting Statistics and Generalized Linear Models

Complete all of the following questions, adding your inputs as code chunks (enclose within triple accent marks) within Rmarkdown.

The exercises are not marked and will not be factored into your course grade, but it is important to complete them to make sure you have the skills to answer assessment questions. You may consult any resource, including other students and the instructor. Please Knit this document to a PDF and upload your work via Canvas at the end of the session. Solutions will be posted for you to check your own answers.

Binomial Simulator

In this section you will implement a random variate generator that directly simulates an event process. If you are unsure how to set up the functions, the functions `threedicerolls` and `many.threedicerolls` from the demonstration correspond to an implementation of a less-general version of the simulator (for p=1/6, N=3 only); you could use this code and generalize it to permit different values of p and N.

1. Write a function that directly simulates N independent tries, each of probability p, and calculates the number of successes. The function can be called `binomialtries`, and should take two arguments: N and p. The function should:
    a. Use R's `sample()` function to generate N random numbers drawn (with replacement) from the two-element vector (0,1) with the probability of zero equal to 1-p and the probability of one equal to p, using `sample()`. Store the result in a vector v.
    b. Calculate k, the number of successes (ones) in v, using `sum()`.
    c. Return the k value.

```
binomialtries = function(N, p) {
  v = sample(0:1, N, prob=c(1-p,p), replace=TRUE)
   k = sum(v==1)
   return(k)
}
```

2. Test the function above by confirming that `binomialtries(10,0)` is always zero, `binomialtries(10,1)` is always 10, and `binomialtries(10,0.5)` varies (typically between 2 and 8).

```
binomialtries(10,1)
```

```
## [1] 10
```

```
binomialtries(10,0.5)
```

```
## [1] 5
```

```
binomialtries(10,0)
```

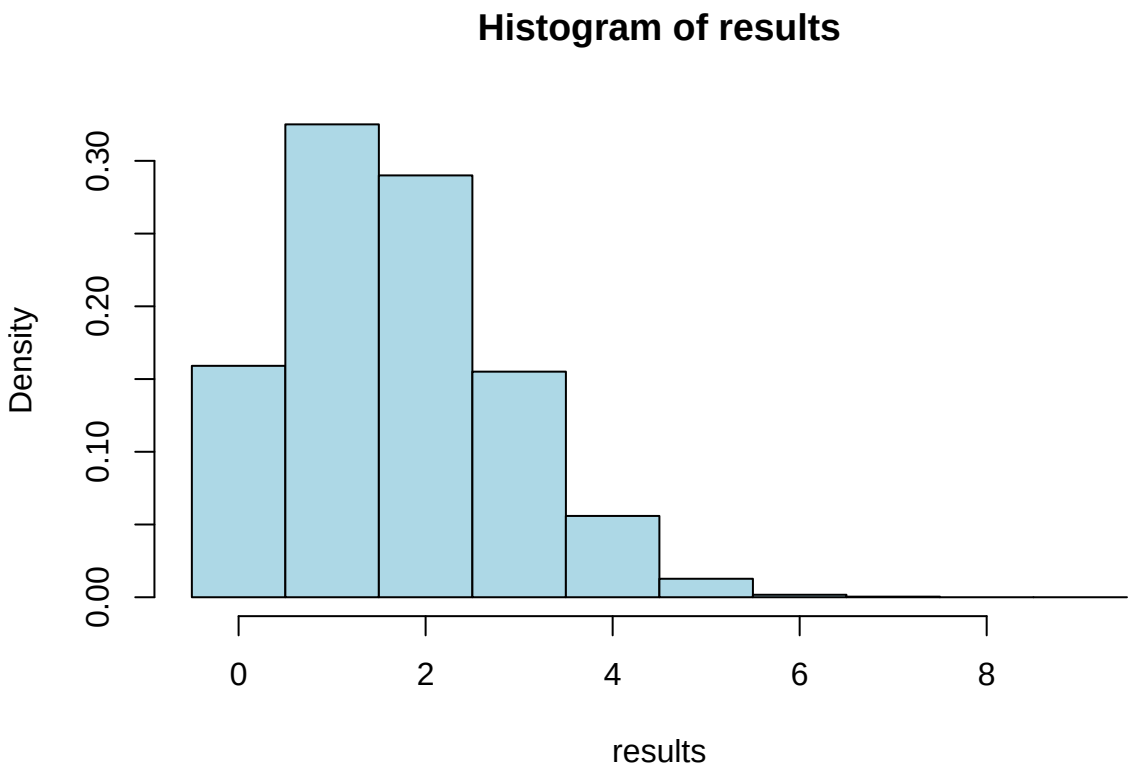
```
## [1] 0
```

3. Write another function that calls `binomialtries` repeatedly, and stores all the returned values in a vector. The function can be called `many.binomialtries`, and should take three arguments: N, p, and nsimulations. (nsimulations is the number of times `many.binomialtries` is called.) The function will need to:
   a. Create an empty vector k of length nsimulations.
   b. Loop over all elements of the vector, calling `binomialtries` each time and storing it in k[i].
   c. Return the k vector.

```
many.binomialtries = function(N, p, nsimulations) {

   k = numeric(nsimulations)
   for (i in 1:nsimulations) k[i] = binomialtries(N, p)
   return(k)
}
```

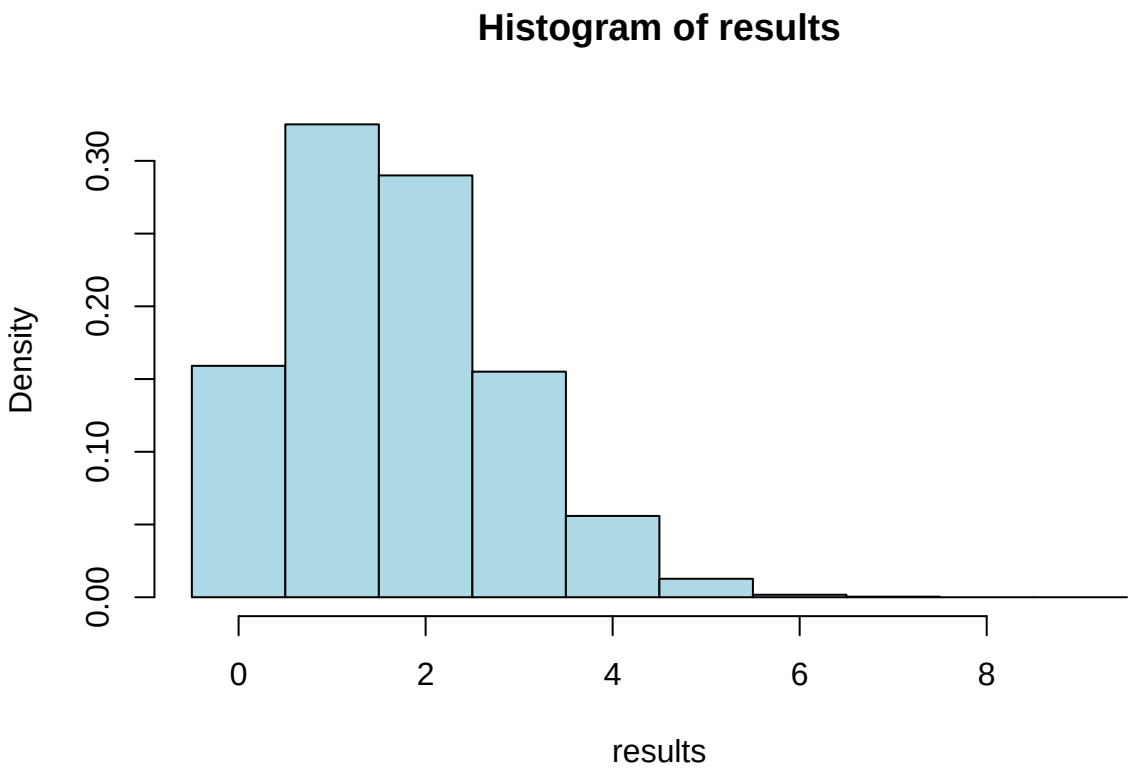
4. Test the function above using N=10 tries, a probability of p=1/6, and a large number of simulations (~10000): this would correspond to rolling ten six-sided dice and measuring how many sixes come up, then repeating that experiment 10000 times. Store the result in a vector and plot a histogram of the results. (Use `freq=FALSE`, and set the breaks to half-integer values to make sure the histogram is informative.)

```
results=many.binomialtries(10, 1/6, 10000)
hist(results, col='lightblue', freq=FALSE, breaks =seq(-0.5,9.5,1) )
```



5. Plot the histogram above again, then overplot the binomial distribution using `dbinom()` (for N=10, p=1/6) to confirm that the two are consistent. (Use `lines()` with `type='b'` to add the binomial distribution.)

```
hist(results, col='lightblue', freq=FALSE, breaks =seq(-0.5,9.5,1) )
```




```
x=0:8
#lines(x, dbinom(xp,size=10,p=1/6), typ='b', lwd=3, cex=1.5)
```

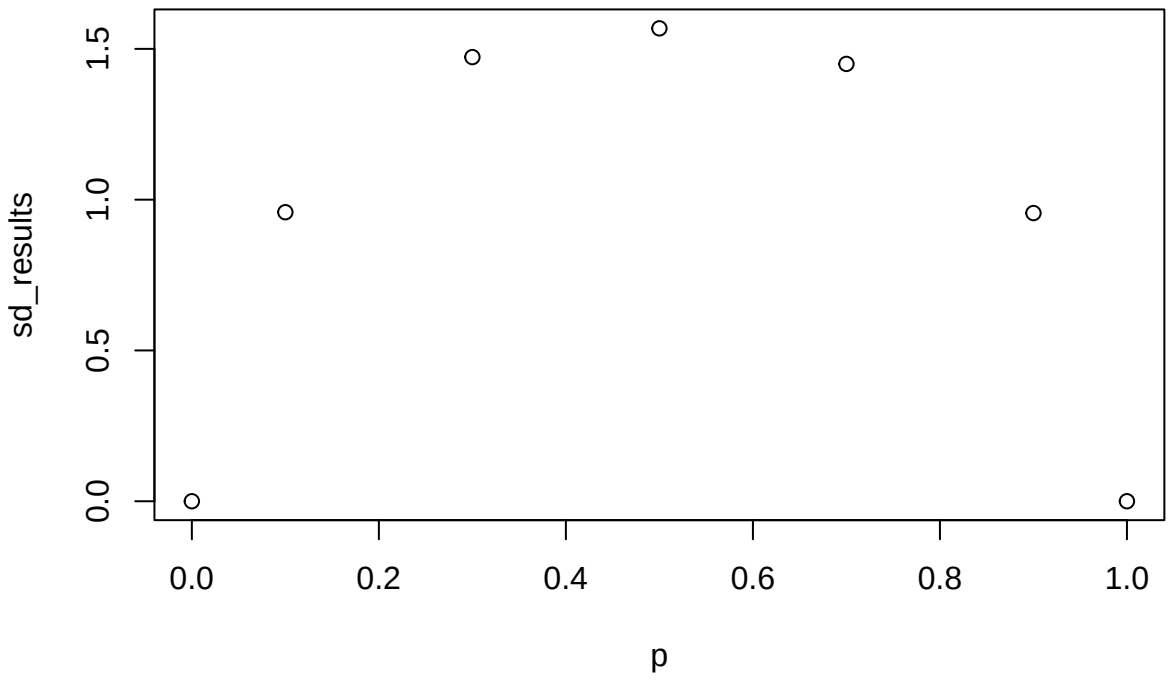
6. Calculate the sample SD of the results from #4.

```
standerror=sd(results)
standerror
```

```
## [1] 1.176124
```

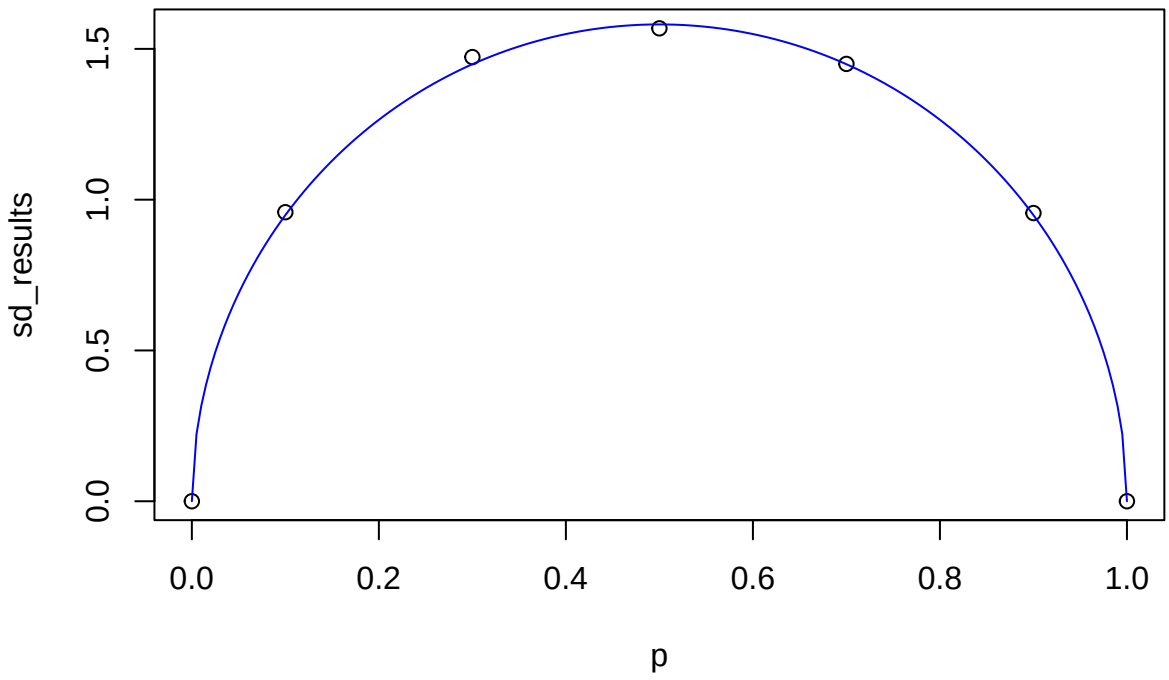
7. Investigate how the sample SD depends on p (continue to assume N=10). Repeat the simulation above for p=0, p=0.1, p=0.3, p=0.5, p=0.7, p=0.9, p=1. Then make a plot of how the SD (y-axis) depends on p (x-axis).

```
n=10
p=c(0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0)
sd_results=numeric(length(p))

for( i in 1:length(p)){
  sd_results[i]=sd(many.binomialtries(n, p[i], 10000))
}

plot(p, sd_results)
```



8. Plot SD versus p (for a binomial distribution with N=10) again using your simulated data, then overplot a curve showing the theoretical standard deviation for binomial data using the equation from lecture (slide #79).

```
plot(p, sd_results)
x=seq(0, 1, 0.005)
lines(x, sqrt(x*(1-x)*n), col='blue')
```



Binomial Simulator in Poisson Limit

9. Now assume that N is large and p is small (Poisson-like or count-like data). Define lambda = pN (i.e., p=lambda/N). Continue using `many.binomialtries` to simulate 10000 random count measurements for lambda=3 (and large N: more than 100, but not *too* large as direct simulations can be computationally expensive). Store the result in a vector.

```
lemda=3
N=300
simulate=many.binomialtries(N, lemda/N, 10000)
```

10. Calculate the SD of the output vector and verify that it is close to sqrt(lambda).

```
sd(simulate)
```

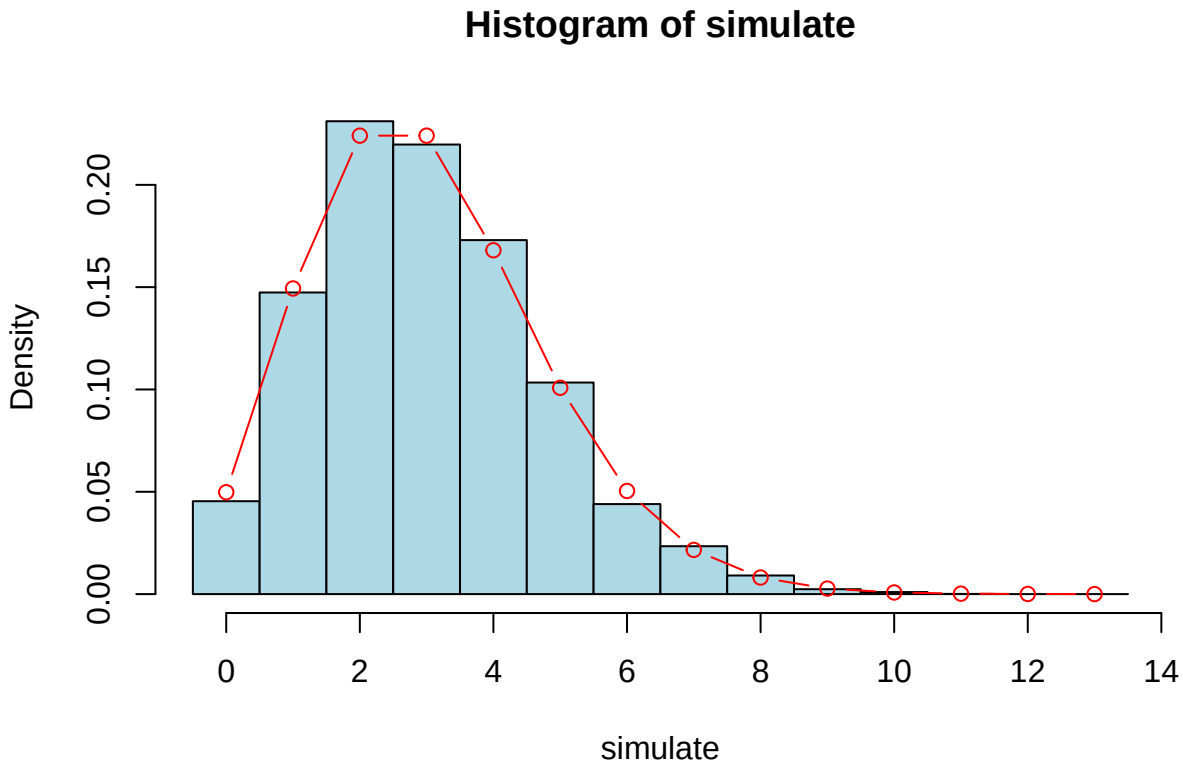
```
## [1] 1.719237
```

```
sqrt(lemda)
```

```
## [1] 1.732051
```

11. Plot a histogram (specify breaks carefully) of the results, using `freq=FALSE`. Overplot the Poisson distribution using `dpois()`, with `type='b'`.

```
hist(simulate, freq=FALSE, breaks=seq(-0.5,13.5,1), col='lightblue')
x=0:13
lines(x, dpois(x, lemda), type='b', col='red')
```



Poisson uncertainties (large lambda)

12. Over the three weeks between 26 Oct and 15 Nov 2022, a total of 535 confirmed cases of coronavirus in Liverpool were reported. Using the simple normal approximation for μ and σ for count-based data (Lecture slide #39) and the "rule of thumb'' for normal confidence intervals (CI = $[\mu - 2\sigma, \mu + 2\sigma]$), and assuming the rate was not changing with time over this period, provide an estimate on the "true" rate of coronavirus per **week**, along with a confidence interval. (Hint: calculate the CI on the rate over a 3-week interval, then divide by 3.)

```
CI = (535 + c(-2,2)*sqrt(535)) / 3
CI
```

```
## [1] 162.9133 193.7534
```

Here are actual counts over these three weeks:

26 Oct to 01 Nov : 165 02 Nov to 08 Nov : 154 09 Nov to 15 Nov : 216

13. The weekly rate calculated in #12 is what we "expect" the weekly count rate to be each week (with random variations from counting statistics) if the rate is not changing with time. Set this value to the variable E, and calculate the vector O-E, the difference between observations and expectations.

```
E = 535/3
O = c(165,154,216)
O-E
```

```
## [1] -13.33333 -24.33333  37.66667
```

14. Calculate Z = (O-E)/sqrt(E).

```
 Z= (O-E)/sqrt(E)
Z
```

```
## [1] -0.9984412 -1.8221551  2.8205963
```

15. Calculate chisquared, the sum of Z^2.

```
chisquared = sum(Z^2)
chisquared
```

```
## [1] 12.2729
```

16. Perform a one-tailed chi-square test using the CDF of the chi-square distribution (`pchisq()` in R) to determine the p-value. Make sure to use the right number of degrees of freedom. Can we conclude that the coronavirus rate in Liverpool has changed significantly with time since the end of October? Is it increasing or decreasing?

```
1-pchisq(chisquared, 2)
```

```
## [1] 0.00216259
```

17. Confirm this result using the convenience function, `chisq.test()`.

```
chisq.test(O)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  O
## X-squared = 12.273, df = 2, p-value = 0.002163
```

Poisson confidence intervals (small lambda)

Now suppose you work at a hospital emergency division and are studying the number of patients admitted during various shifts. On one particular day you measure the following: morning: 5 patients admitted
afternoon: 7 patients admitted
evening: 8 patients admitted
night: 0 patients admitted

Assume that each patient-event is independent.

18. Calculate the precise 95% confidence interval for the average rate of daily patient arrivals for each of these four windows. Use the chi-squared distribution quantile function solution as given in lecture (slide #46).

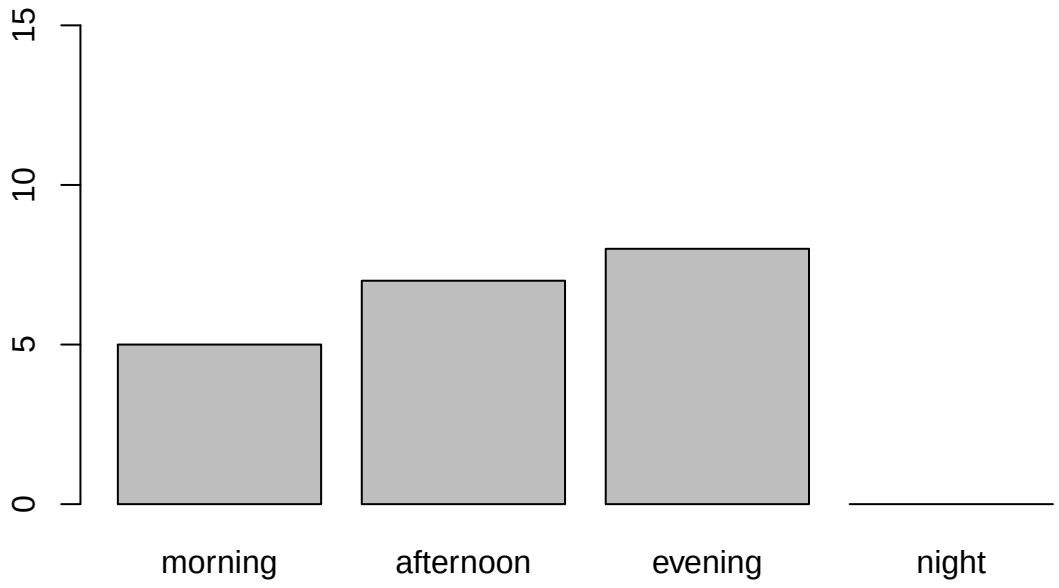
```
patients = c(5,7,8,0)
conf_lower = qchisq(0.025, 2*patients)/2
conf_upper = qchisq(0.975, 2*(patients+1))/2
conf_lower;conf_upper
```

```
## [1] 1.623486 2.814363 3.453832 0.000000
```

```
## [1] 11.668332 14.422675 15.763189  3.688879
```

19. Make a plot summarising the above information: a bar plot to give the observed counts, with error bars added to indicate the upper and lower limits on the 95% confidence interval. (Use `barplot()` to draw the plot; add error bars with `arrows` or use the `errorbars` function from Week 4. You can label the bars by making the vector of patient counts into a named vector: e.g. `names(v) = c('morning','afternoon',...)`)

```
names(patients) = c('morning','afternoon','evening','night')
pos = barplot(patients,ylim=c(0,16))
```




```
errorbars = function(x, ymin, ymax) arrows(x,ymin,x,ymax,code=3,angle=90)
#errorbars(pos, confintlower, confintupper)
```

20. Perform a chi-squared contingency test on the patient numbers to test the null hypothesis that the rate of patients needing treatment is constant over the day. Is the p-value exact or approximate?

```
chisq.test(patients)
```

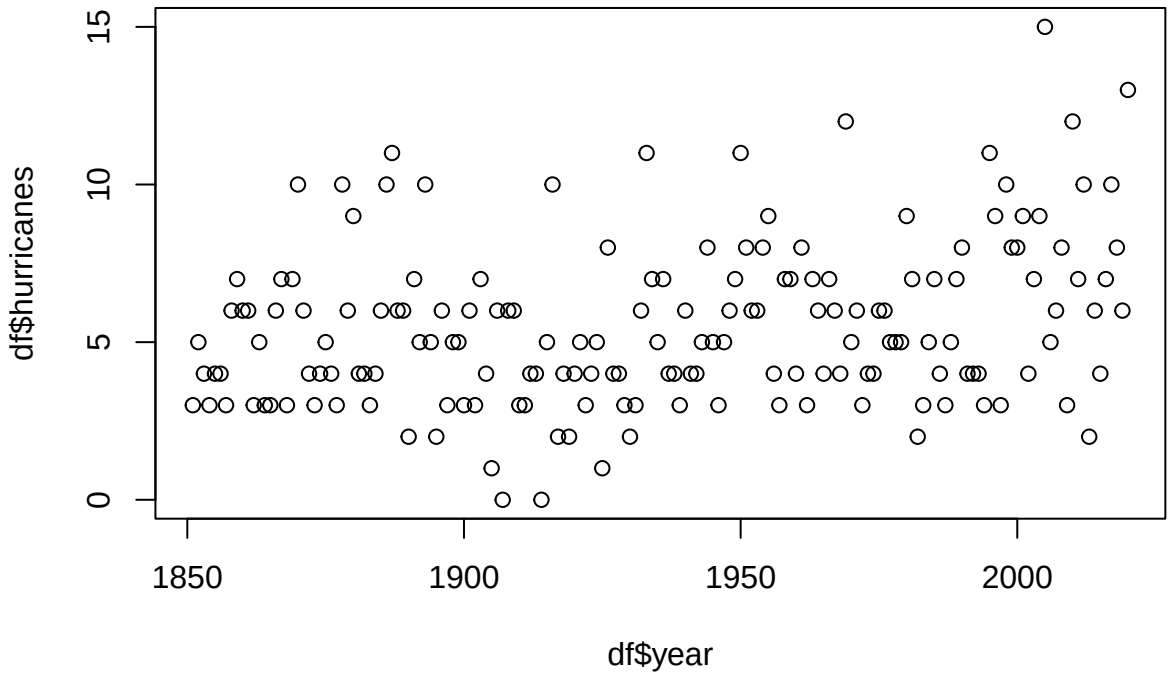
```
##
##  Chi-squared test for given probabilities
##
## data:  patients
## X-squared = 7.6, df = 3, p-value = 0.05504
```

Count Regression

The file hurricane.csv contains data on the number of hurricanes recorded in the North Atlantic each year since 1851 (source: NOAA). The categories are: year: Calendar year storms: Number of tropical cyclonic storms hurricanes: Number of hurricanes (winds > 74 mph) severe: Number of severe hurricanes (winds > 110 mph) ACE: Accumulated cyclone energy

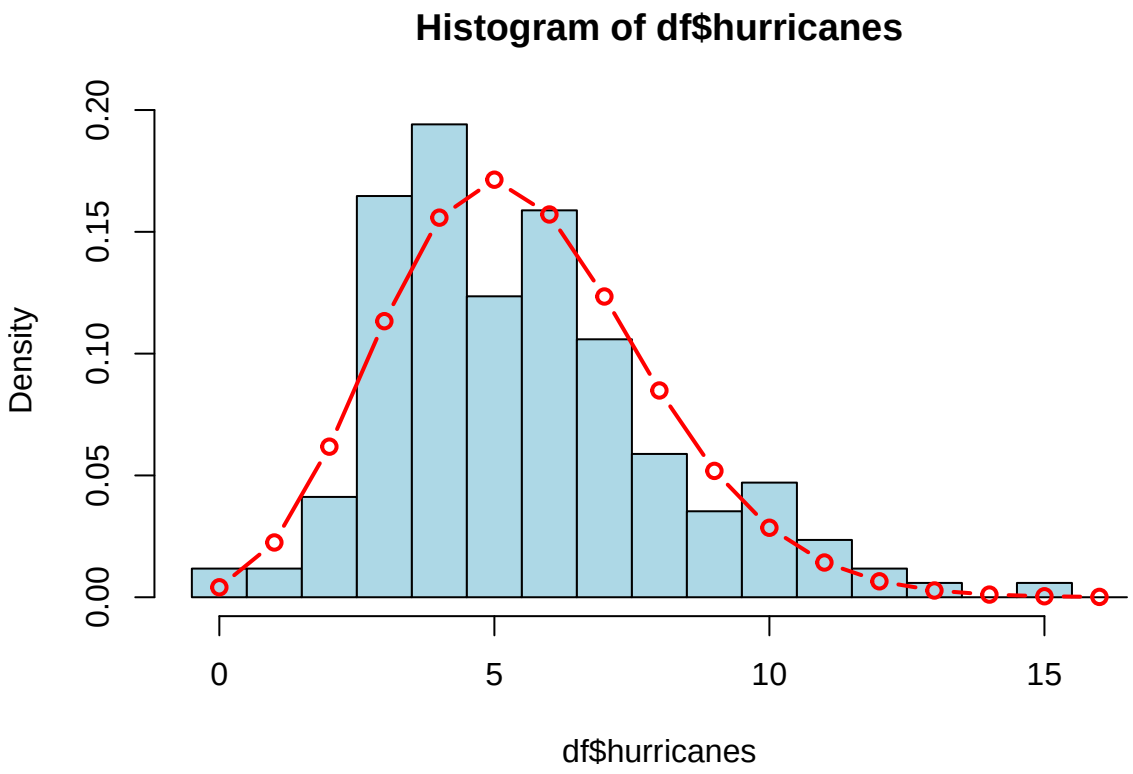
21. Read the data in from disk and make a scatter plot of the number of hurricanes (y-axis) each year (x-axis).

```r
df=read.csv('hurricane.csv')
#head(df)
plot(df$year, df$hurricanes)
```



22. Make a histogram showing the distribution of yearly hurricane counts. (As always, be careful with the breaks if you use the `hist()` function.) Overplot a Poisson distribution curve with the mean (rate) parameter set equal to the mean number of hurricanes per year. Do the data appear to be consistent with a model in which the events are independent and the rate is unchanging?

```r
x = 0:16
hist(df$hurricanes,breaks=seq(-0.5,16.5,1),col='lightblue', freq=FALSE)
lines(x,dpois(x, mean(df$hurricanes)),lwd=2,typ='b', col='red')
```



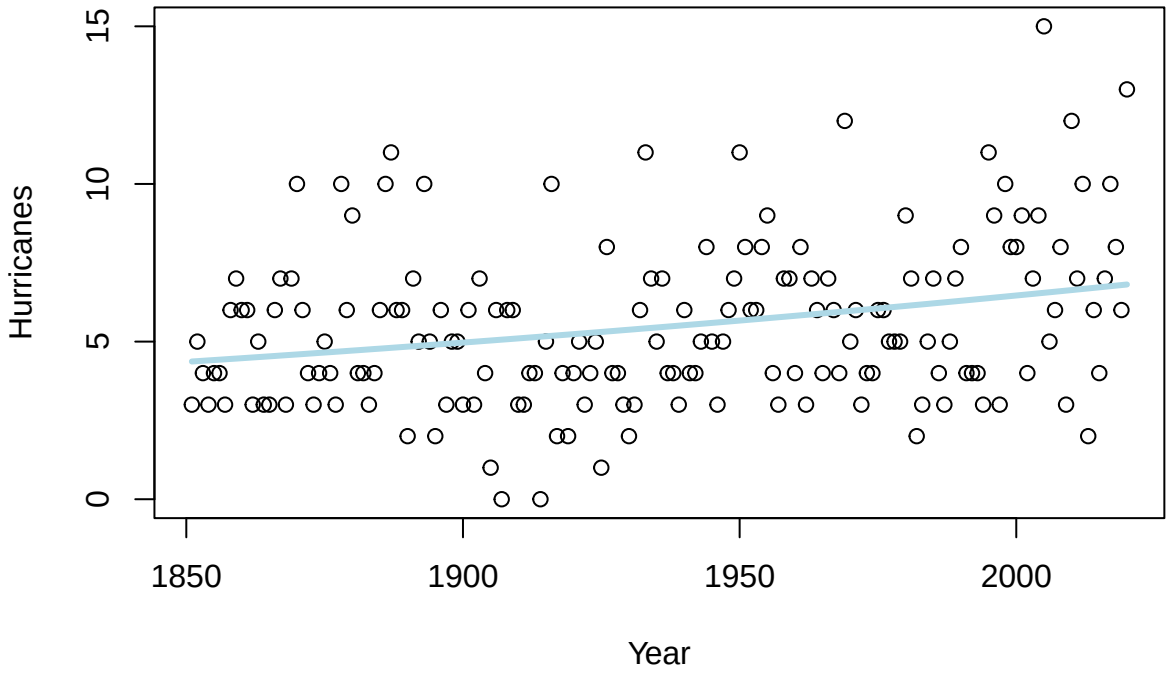
23. Use a generalized linear model to fit the yearly hurricane numbers as a function of year. Does this suggest that the rising trend is significant?

```
glm_model = glm(hurricanes~year,data=df,family='poisson')
summary(glm_model)
```

```
##
## Call:
## glm(formula = hurricanes ~ year, family = "poisson", data = df)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.3896902  1.3009027  -2.606  0.00917 **
## year         0.0026278  0.0006697   3.924 8.72e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 207.60  on 169  degrees of freedom
## Residual deviance: 192.13  on 168  degrees of freedom
## AIC: 781.96
##
## Number of Fisher Scoring iterations: 4
```

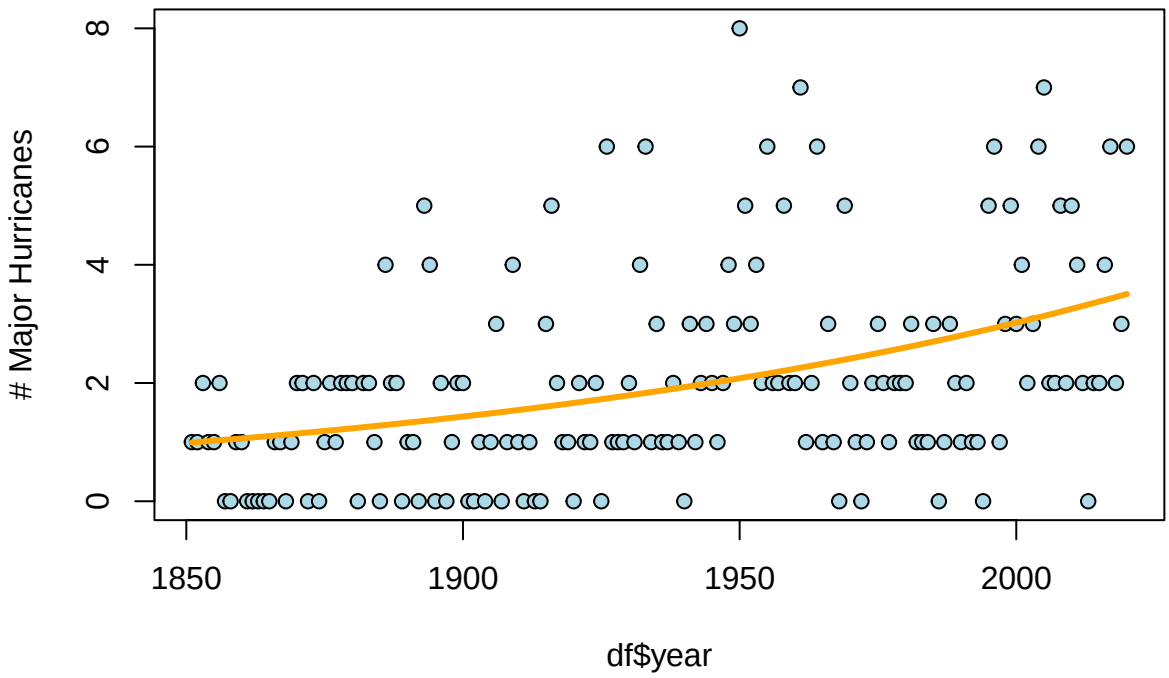
24. Plot the model function on top of the hurricane data as a coloured line, using `predict()`. (Be sure to specify `type='response'`!) Optional: also add a confidence interval on the model curve.

```
plot(df$year, df$hurr, bg='lightblue', xlab='Year', ylab='Hurricanes')
lines(df$year, predict(glm_model,type='response'), col='lightblue', lwd=3)
```



25. Repeat #23 and #24 for major hurricanes.

```
glm_model2 = glm(severe~year,data=df,family='poisson')
plot(df$year, df$severe, pch=21, bg='lightblue', ylab='# Major Hurricanes')
lines(df$year, predict(glm_model2,type='response'), col='orange', lwd=3)
```



Logistic Regression

You are conducting an opinion poll on some hot-button issue of the day. Your company rings 300 people and records their ages, genders, annual incomes, religious views (0=not religious; 1=somewhat religious; 2=very religious), and whether they agree or disagree with an issue statement. The data file is opinionpoll.csv.

26. Read in the data from disk. How many of the respondents "agree", and how many "disagree"?

```
poll = read.csv('opinionpoll.csv')
table(poll$issue)
```

```
##
##    agree disagree
##      166      134
```

27. What proportion agree, and what proportion disagree?

```
table(poll$issue)/nrow(poll)
```

```
##
##     agree  disagree
## 0.5533333 0.4466667
```

28. What is the odds ratio for agreement vs. disagreement?

```
t = unname(table(poll$issue))
t[2]/t[1]
```

```
## [1] 0.8072289
```

29. Suppose that the true population proportion were 0.50 (i.e., an infinitely large poll would obtain a perfect 50/50 split). What is the probability of your survey providing the fraction you actually observed, or an equal-or-more-extreme fraction? (You may use the convenience functions.)

    a. Use a chi-square test.

    b. Use a binomial test.

```
chisq.test(t)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  t
## X-squared = 3.4133, df = 1, p-value = 0.06467
```

```
binom.test(t)
```

```
##
##  Exact binomial test
##
## data:  t
## number of successes = 166, number of trials = 300, p-value = 0.07331
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.4951193 0.6104837
## sample estimates:
## probability of success
##              0.5533333
```

30. Assume that your survey respondents are a random sampling of the country as a whole. Based on your result above, does your survey provide evidence that a majority of people "agree"?

Write your response here.

31. Quote a 95% confidence interval on the population proportion that agrees.

```
binom.test(t)$conf.int[1:2]
```

```
## [1] 0.4951193 0.6104837
```

32. Carry out a logistic regression using `glm()`. You may ignore interaction terms. Which of the four explanatory variables appear to be significant predictors of the respondent's view on the poll question?

```r
agree = poll$issue=='agree'
glm_poll= glm(agree~age+gender+income+religion, family='binomial', data=poll)
summary(glm_poll)
```

```
##
## Call:
## glm(formula = agree ~ age + gender + income + religion, family = "binomial",
##     data = poll)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.691e+00  3.624e-01   4.665 3.09e-06 ***
## age         -2.260e-02  6.961e-03  -3.247  0.00117 **
## genderM      2.402e-01  2.472e-01   0.972  0.33115
## income      -4.644e-06  3.401e-06  -1.366  0.17209
## religion    -3.215e-01  1.545e-01  -2.082  0.03738 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 412.47  on 299  degrees of freedom
## Residual deviance: 385.17  on 295  degrees of freedom
## AIC: 395.17
##
## Number of Fisher Scoring iterations: 4
```

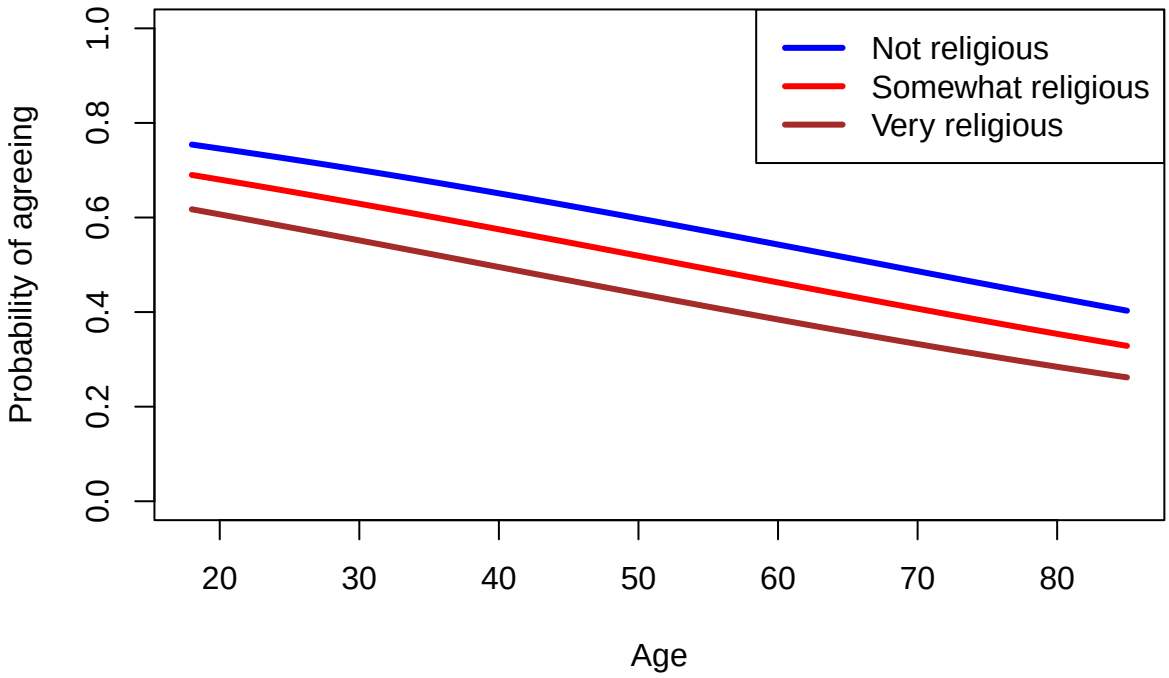
33. According to your model, what is the probability that a randomly-chosen 55-year-old female recipient, with an income of 35,000 and no religious beliefs, "agrees" with the poll statement?

```r
predict(glm_poll,list(age=55,gender='F',income=35000,religion=0),type='response')
```

```
##         1
## 0.5707609
```

34. Make a plot of the model probability of "agreeing" as a function of age (x-axis variable) and religion (colour), controlling for other factors by holding the other terms constant at "typical" values (e.g., their median).

```r
ageplot = 18:85
np = length(ageplot)
plot(0,0,xlim=c(18,85),ylim=c(0,1),typ='n',xlab='Age', ylab='Probability of agreeing')
lines(ageplot, predict(glm_poll,list(age=ageplot,gender=rep('F',np),income=rep(35000,np),
religion=rep(0,np)),type='response'),typ='l',lwd=3,col='blue')


lines(ageplot, predict(glm_poll,list(age=ageplot,gender=rep('F',np),income=rep(35000,np),
religion=rep(1,np)),type='response'), typ='l',lwd=3,col='red')
lines(ageplot, predict(glm_poll,list(age=ageplot,gender=rep('F',np),income=rep(35000,np),
religion=rep(2,np)),type='response'), typ='l',lwd=3,col='brown')
legend('topright',c('Not religious','Somewhat religious', 'Very religious'),
col=c('blue','red','brown'), lty=c(1,1),lwd=c(3,3))
```


Probability of agreeing
0.0
0.2
0.4
0.6
0.8
1.0
20
30
40
50
60
70
80
Age
Not religious
Somewhat religious
Very religious