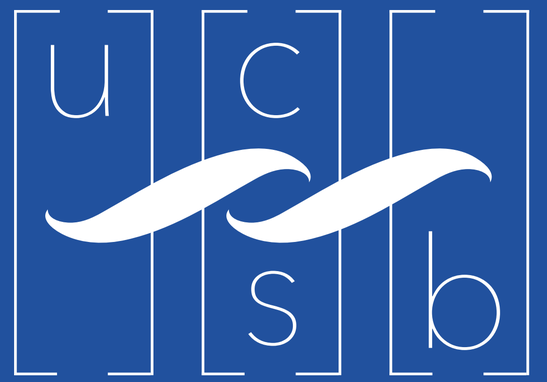
Study of Movie Interest Over Time

Alex Dorsett

Thursday, January 14, 2021

MAT259



Project Overview

- Want to study movie popularity as a function of time
  - How long does interest in a movie last after its release?
  - Does interest drop-off exponentially?
  - Does box-office success indicate there will be more interest in a movie?
- Will count how many times a movie is checked each month
  - Look at how this number changes over time
- Compare this checkout rate between different categories of movies
  - Only looking at movies that were released within the time period the data is available
- Will accomplish this with MySQL, python, and ROOT

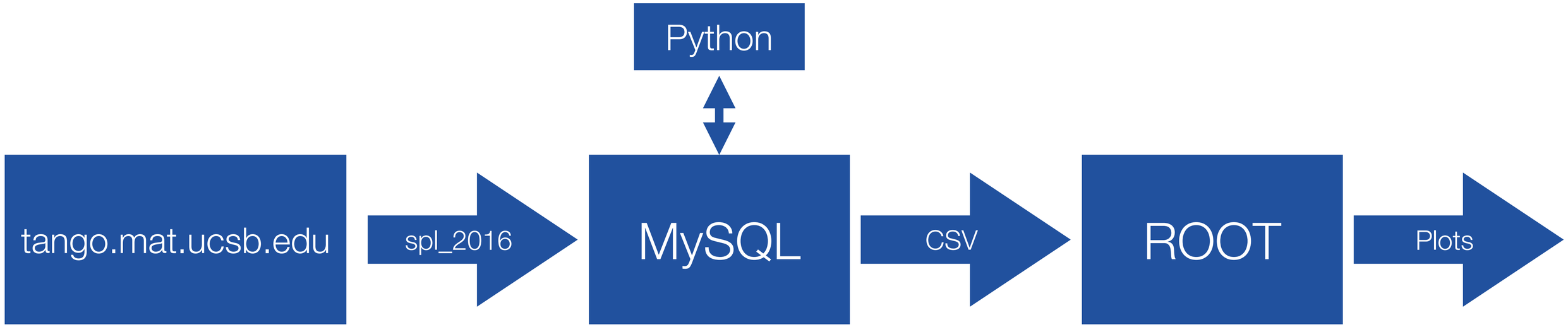
Movie selection

- Requirements
  - Released for public rental between 2006-2016
- Movie selection is divided into 2 categories
  - Critically-acclaimed
  - Blockbusters
- Critically-acclaimed
  - Winners of academy award for best picture from 2007-2016
- Blockbusters
  - Collection of films which performed well in the domestic box office from 2007-2016
  - Tried to include mix of genres and intended audiences
  - Release dates distributed evenly across time
- Thought about studying the effect of a sequel being released, but didn't have good enough statistics

Critically-Acclaimed	Blockbusters
The Departed	The Dark Knight
No Country For Old Men	Avatar
Slumdog Millionaire	Toy Story 3
The Hurt Locker	The Avengers
The King's Speech	Skyfall
The Artist	Gravity
Argo	Frozen
12 Years a Slave	The LEGO movie
Birdman	American Sniper
Spotlight	Star Wars Episode VII

Data pipeline

- Queried spl_2016 database with SQL
  - Seattle Public Library data from 2006-present
- The code gets really long when I want to get all the data at once
  - For flexibility, I wrote a python script that generates the necessary SQL code for a list of movies I provide
- For the analysis and plotting of the data, I used a package based in C++
  - Put the data into histograms, and then fit them with an exponential distribution



SQL Query

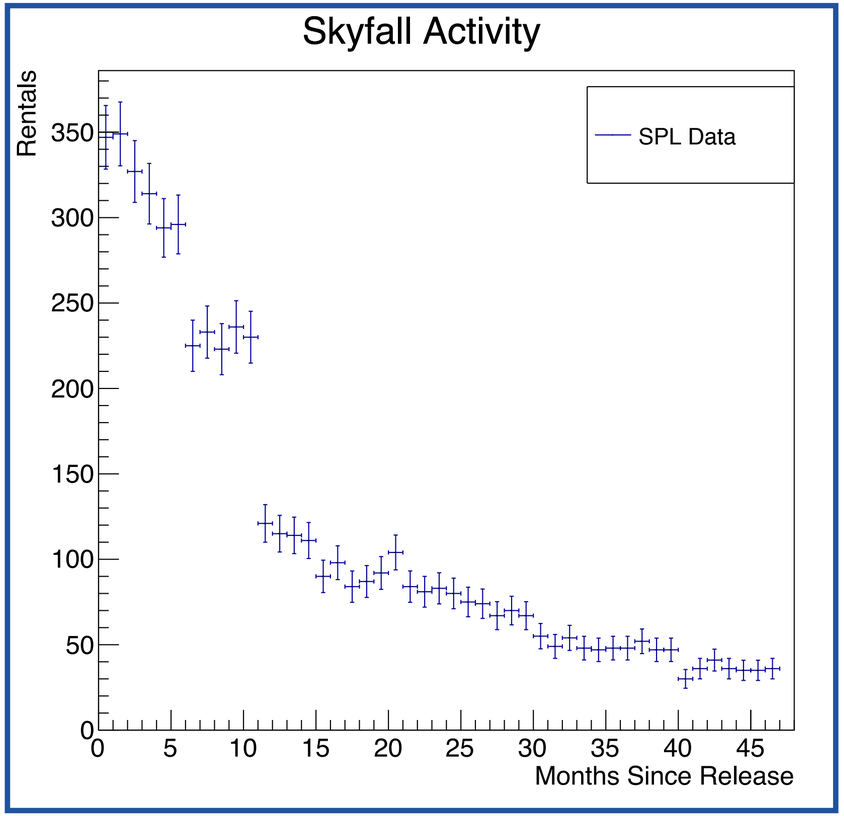
```sql
SET @bibNum1 = 2647147; -- Store bibNumbers as variables
SET @bibNum2 = 2871394;
SET @bibNum3 = 2398486;
SELECT @RelDate1:=MIN(cout) FROM spl_2016.outraw WHERE bibNumber = @bibNum1; -- Find earliest
SELECT @RelDate2:=MIN(cout) FROM spl_2016.outraw WHERE bibNumber = @bibNum2; -- checkout date
SELECT @RelDate3:=MIN(cout) FROM spl_2016.outraw WHERE bibNumber = @bibNum3; -- for each movie
SELECT
    CASE -- Calculate MonthsSinceRelease separately for each movie in the first row
      WHEN bibNumber = @bibNum1 THEN FLOOR(DATEDIFF(cout, @RelDate1) / 28)
      WHEN bibNumber = @bibNum2 THEN FLOOR(DATEDIFF(cout, @RelDate2) / 28)
      WHEN bibNumber = @bibNum3 THEN FLOOR(DATEDIFF(cout, @RelDate3) / 28)
    END AS MonthsSinceRelease,
    -- Count how many movies with a given bibNumber were checked out each month (28 days)
    COUNT(IF(bibNumber = @bibNum1, FLOOR(DATEDIFF(cout,@RelDate1)/28),NULL)) as Avatar,
    COUNT(IF(bibNumber = @bibNum2, FLOOR(DATEDIFF(cout,@RelDate2)/28),NULL)) as Skyfall,
    COUNT(IF(bibNumber = @bibNum3, FLOOR(DATEDIFF(cout,@RelDate3)/28),NULL)) as 'The Departed'
FROM
    spl_2016.outraw
WHERE -- only include movies being counted
    bibNumber = @bibNum3 or
    bibNumber = @bibNum2 or
    bibNumber = @bibNum1
GROUP BY 1 -- Group+order by MonthsSinceRelease
ORDER BY 1
LIMIT 48; -- Incluce 48 months of data
```

- Initially had trouble getting data for multiple movies with a single query
  - Wanted to get the data as a function of “months since release”
  - Each movie has a different release date, so the first column of data has to be defined differently for each movie
- Eventually figured out how to accomplish this with a case statement
  - Here is an example of a query that gets the data for 3 movies
  - Wrote python script to generate the code for the 20-movie query
- Database data used
  - outraw: cout, bibNumber, title

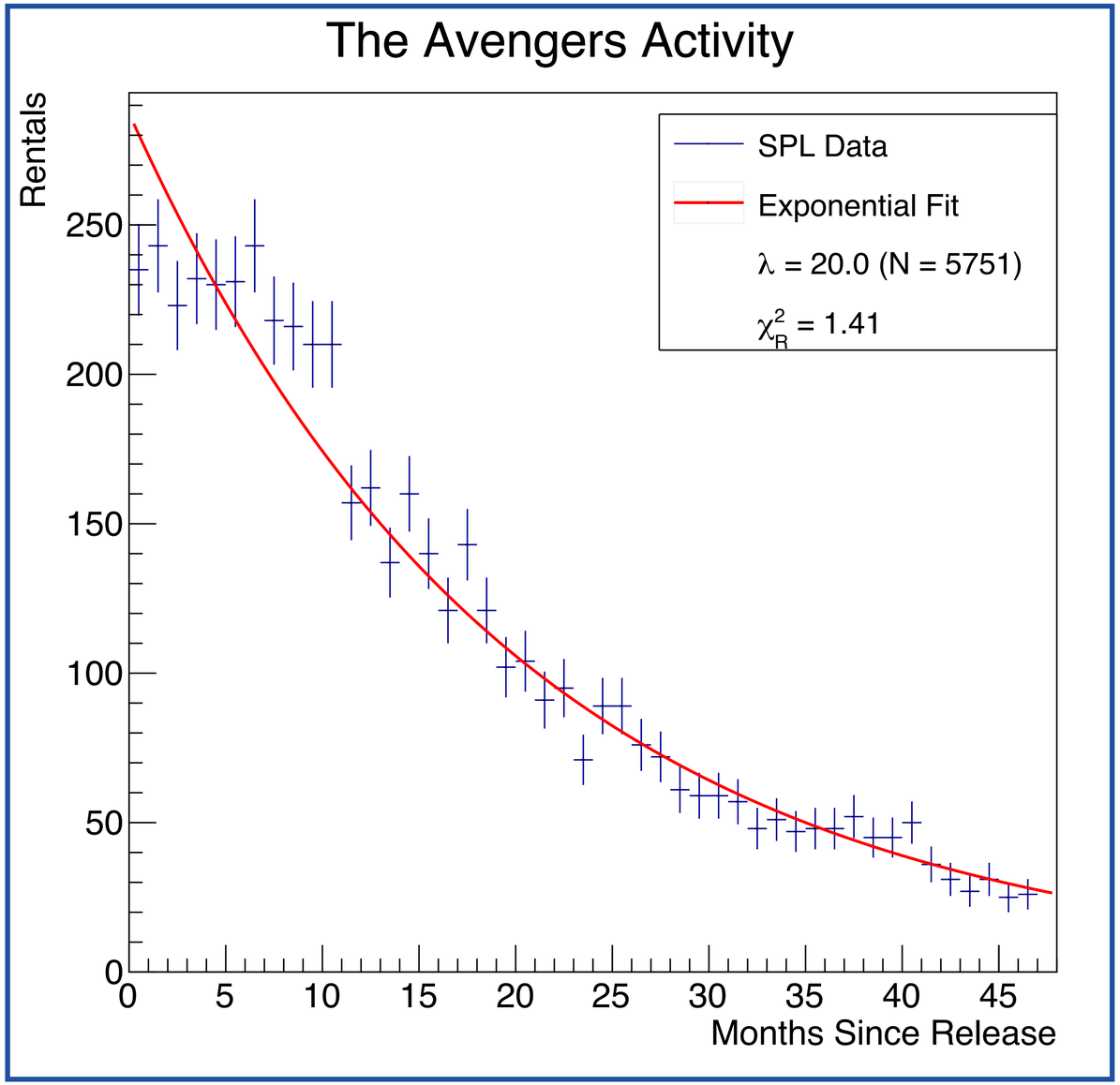
Resulting data

MonthsSinceRelease	The Depart	No Country	Slumdog M	The Hurt Lo	The King's S	The Artist	Argo	12 Years a S	Birdman	Spotlight	The Dark Kı	Avatar	Toy Story 3	The Avenge	Skyfall	Gravity	Frozen	The LEGO r	American S	Star Wars E
0	127	266	381	334	280	279	380	313	336	425	376	376	283	303	311	432	259	247	341	339
1	161	498	388	316	223	203	510	475	468	405	411	274	192	235	347	373	302	230	256	258
2	113	470	396	306	193	115	554	451	423	372	410	273	208	243	349	392	318	249	253	255
3	162	438	432	270	209	203	518	411	414	375	386	266	194	223	327	389	328	246	273	247
4	165	476	434	270	157	203	503	397	423	344	338	162	197	232	314	365	298	229	233	228
5	146	509	263	289	202	205	482	386	397	333	366	221	183	230	294	349	284	221	211	224
6	131	486	405	261	196	196	358	395	365	330	445	245	195	231	296	347	286	209	232	229
7	112	449	496	286	178	240	427	357	289	234	391	230	175	243	225	338	294	239	261	255
8	119	390	484	163	170	225	354	278	238	238	353	276	181	218	233	325	292	199	203	230
9	145	452	421	309	169	246	349	298	340	198	283	246	189	216	223	291	309	200	214	219
10	151	428	474	346	163	244	185	257	276	197	323	260	121	210	236	263	251	204	213	232
11	140	422	429	288	192	199	165	181	305	156	386	273	140	210	230	226	260	158	194	136
12	162	426	428	275	189	195	121	162	223	97	329	229	127	157	121	219	239	169	145	150
13	166	355	433	257	229	195	127	159	276	110	323	173	126	162	115	213	238	161	149	166
14	180	374	401	246	184	162	113	143	209	124	333	177	131	137	114	204	217	181	142	153
15	173	343	368	236	207	147	120	132	147	87	305	161	88	160	111	181	212	169	111	146
16	193	377	364	200	208	121	108	119	141	78	273	169	98	140	90	159	203	140	114	150
17	170	382	346	180	149	120	116	129	119	95	287	102	82	121	98	153	206	133	94	110
18	184	348	256	156	208	112	109	123	116	58	282	146	63	143	84	155	200	133	92	89
19	163	220	300	161	194	114	102	108	107	60	247	137	76	121	87	144	179	127	81	105
20	151	333	334	136	192	110	115	98	113	50	258	106	82	102	92	136	181	110	82	104
21	124	319	320	71	177	98	94	103	100	46	224	108	99	104	104	141	171	106	57	93
22	150	327	317	102	167	80	96	109	108	48	151	101	75	91	84	127	145	110	53	101
23	118	299	328	95	171	80	101	101	95	43	183	112	63	95	81	120	151	108	51	0
24	117	287	276	90	175	71	85	99	111	0	188	103	74	71	83	119	130	76	45	0
25	115	277	286	80	144	67	79	115	80	0	149	103	62	89	80	108	127	81	45	29
26	120	279	255	88	118	59	78	104	88	23	160	110	69	89	75	131	129	95	34	88
27	91	214	232	78	109	56	64	95	94	46	158	95	68	76	74	92	109	98	45	78
28	113	238	195	86	105	55	73	76	69	29	161	104	71	72	67	89	107	92	40	71
29	111	223	207	92	114	45	53	86	65	36	146	93	71	61	70	71	102	71	28	63
30	99	181	199	58	99	46	54	67	60	38	142	75	67	59	67	75	113	62	30	57
31	95	187	143	67	98	56	53	72	55	21	118	88	55	59	55	66	87	64	28	8
32	58	133	174	79	107	41	66	65	50	0	137	89	57	57	49	47	97	59	31	23
33	104	173	146	77	95	54	52	69	41	19	138	94	50	48	54	67	72	45	27	26
34	96	160	157	43	92	48	46	72	37	21	102	76	57	51	48	62	91	41	10	29
35	80	164	158	66	86	37	48	75	36	27	72	95	53	47	47	65	77	51	0	37
36	82	166	148	64	85	36	43	62	27	31	99	85	47	48	48	52	70	44	0	34
37	79	136	167	53	102	31	55	50	10	36	91	96	43	48	48	66	59	38	13	26
38	67	141	158	47	92	31	53	35	0	27	99	86	33	52	52	49	58	37	13	25
39	87	137	144	57	79	33	50	51	0	24	76	73	36	45	47	51	62	40	12	21
40	74	131	111	54	85	25	52	43	14	18	65	70	29	45	47	47	47	39	10	21
41	74	131	110	46	86	26	46	37	26	15	69	82	37	50	30	34	49	33	11	19
42	75	135	132	51	69	30	36	27	8	20	75	77	29	36	36	36	41	24	5	15
43	66	110	137	42	68	28	36	36	14	13	81	77	31	31	41	36	42	22	0	12
44	53	109	86	42	71	27	36	27	10	14	67	77	25	27	36	30	37	19	8	14
45	35	73	92	43	65	27	36	22	11	11	51	69	24	31	35	27	37	12	7	12
46	60	91	126	35	60	22	33	22	0	13	63	53	22	25	35	22	31	6	5	13
47	61	96	105	39	58	32	27	18	7	16	59	52	23	26	36	28	24	0	4	13

- 48 data points for each movie
  - (Months since release, # checkouts)
- Easier to study when plotted



Analysis Details

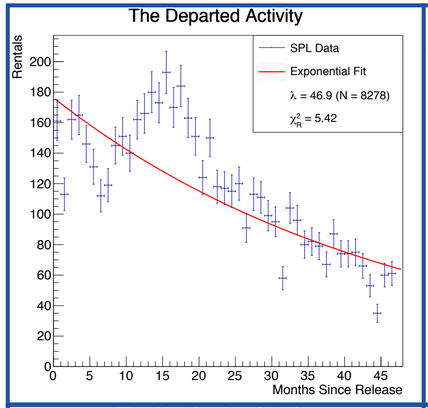


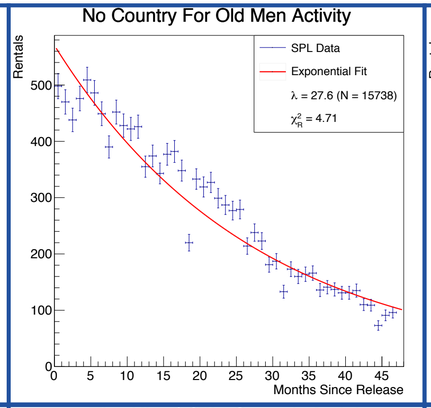
- Fit data with an exponential distribution

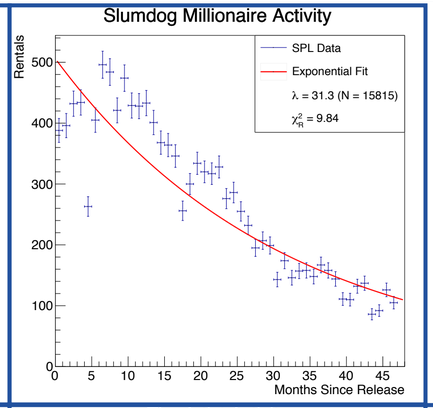
$$f(t) = Ne^{-\frac{t}{\lambda}}$$

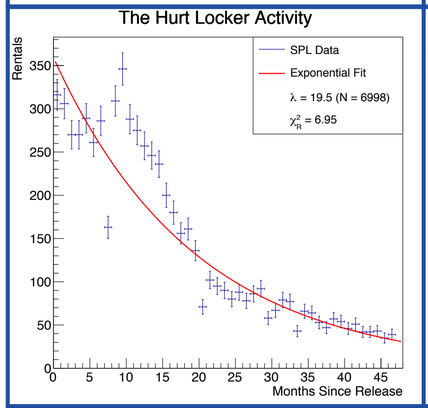
- For data, Poisson errors are assumed
  - Error on X rentals/month = $\sqrt{X}$
- Interpret best-fit λ as lifetime
- Also examine $\chi_R^2 = \frac{1}{n_b - 1}\sum_i \frac{(O_i - E_i)^2}{\sigma_i^2}$
  - O_i = observed value, σ_i = uncertainty in observed value
  - E_i = expected value (from fit function)
  - n_b = number of bins
- χ_R^2 value near 1 indicates fit matches data well
  - See if fit performs well for each category

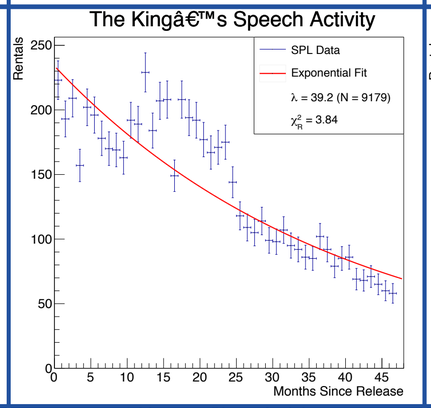
Results: Critically-acclaimed movies

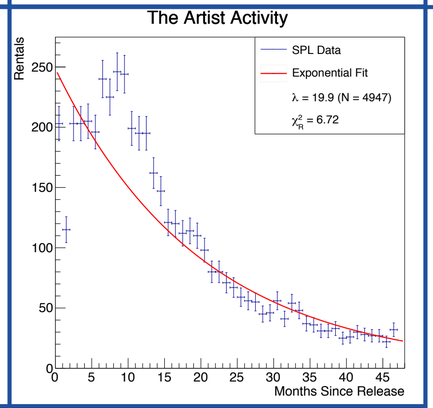


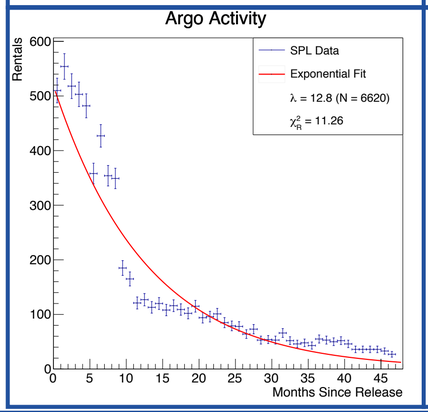


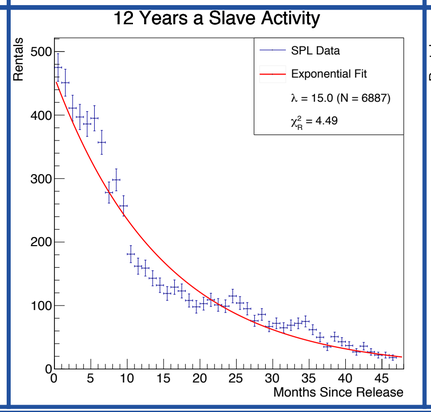


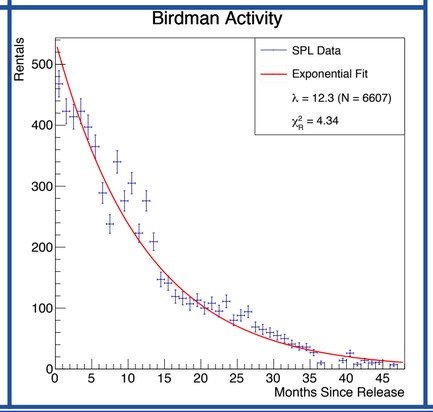




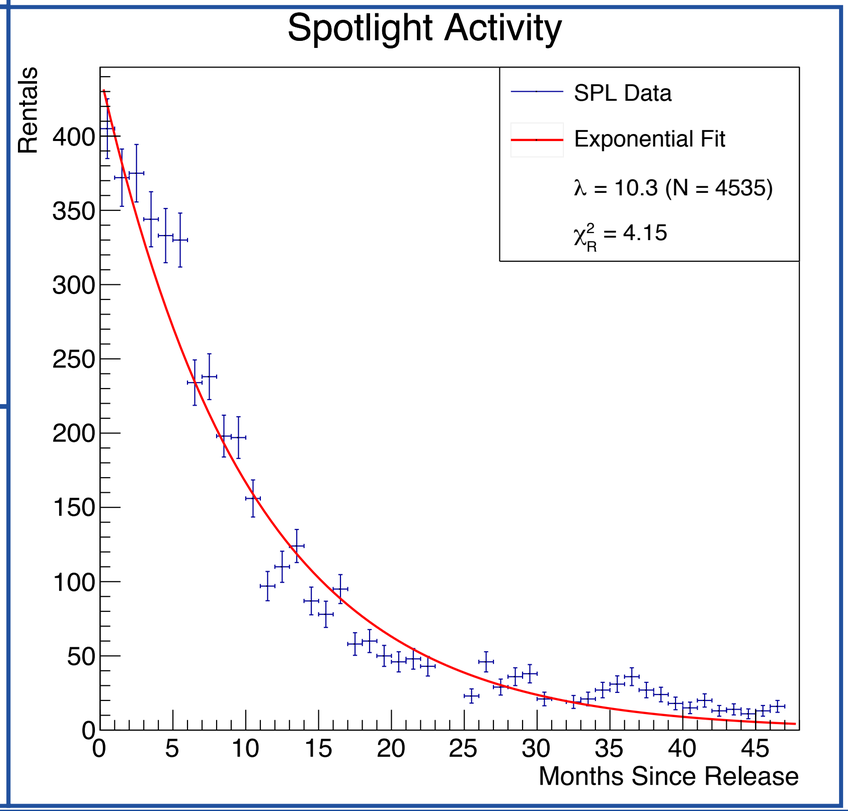




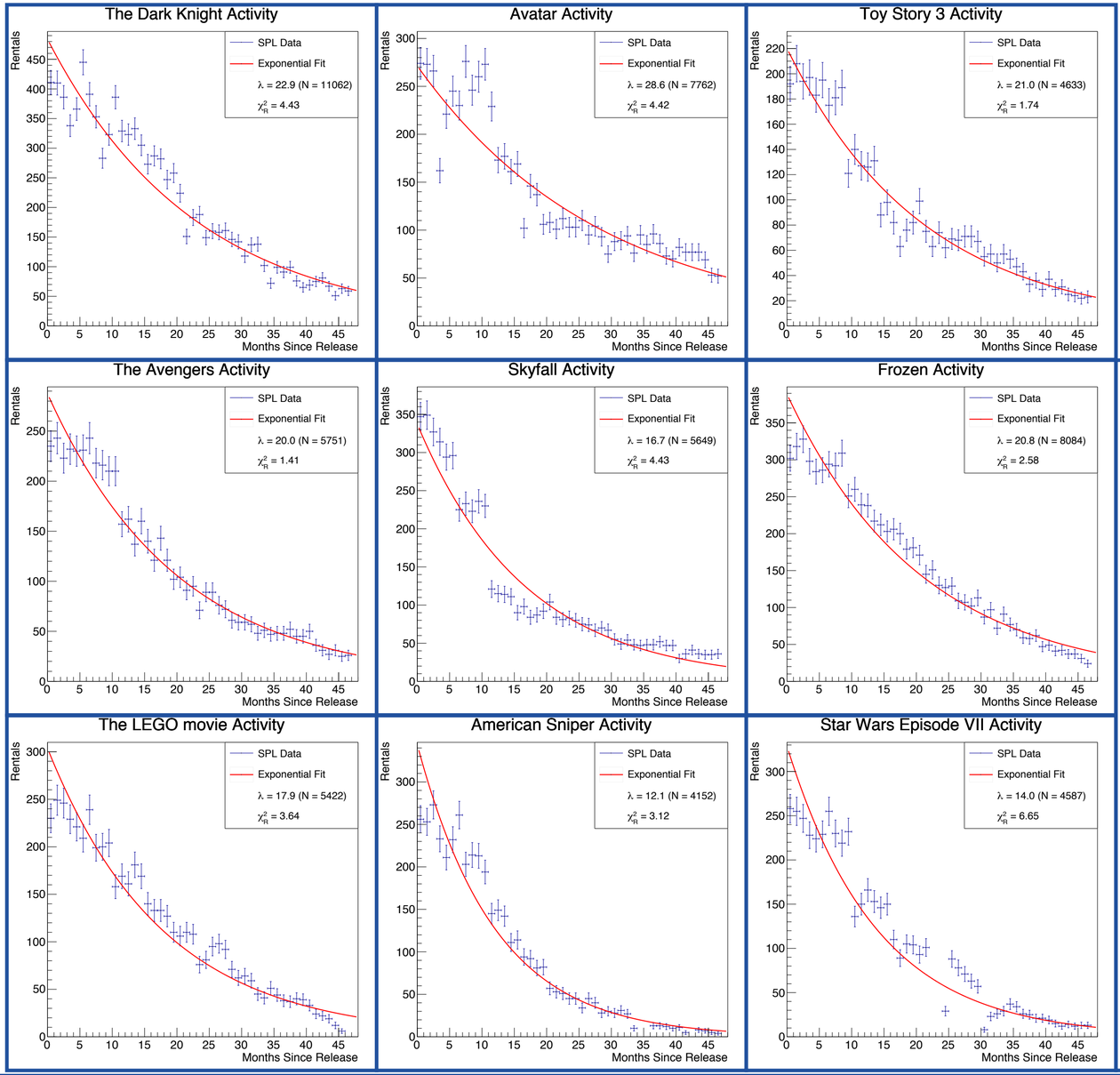




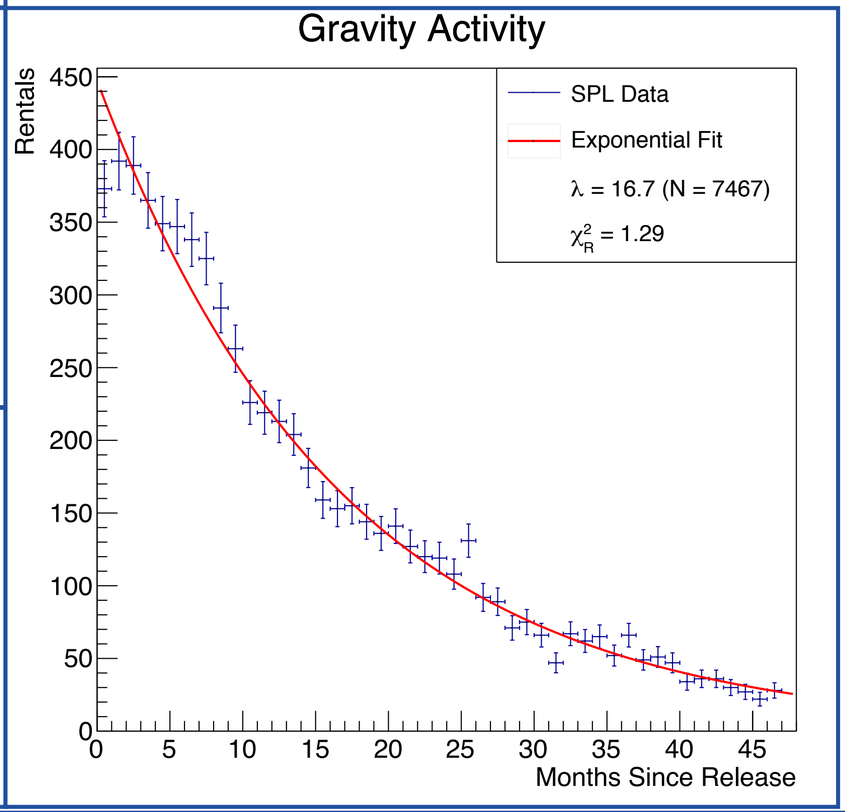
- Showing data+fit results
- Overall, fits don’t look that great…
  - Movies have interest bumps a year or more after release
  - Even for movies with monotonically decreasing activity, drop-off doesn’t seem to be exponential in nature



Results: Blockbusters



- Showing data+fit results
- Fits look better here
  - Though some fits perform very well, overall behavior doesn't look exponential



Category comparison

Critically-Acclaimed			
Movie	λ	N	χ^2
The Departed	46.9	8278	5.42
No Country For Old Men	27.6	15738	4.71
Slumdog Millionaire	31.3	15815	9.84
The Hurt Locker	19.5	6998	6.95
The King's Speech	39.2	9179	3.84
The Artist	19.9	4947	6.72
Argo	12.8	6620	11.26
12 Years a Slave	15.0	6887	4.49
Birdman	12.3	6607	4.34
Spotlight	10.3	4535	4.15
Average	**23.5**	**8560**	**6.2**

Blockbusters			
Movie	λ	N	χ^2
The Dark Knight	22.9	11062	4.43
Avatar	28.6	7762	4.42
Toy Story 3	21.0	4633	1.74
The Avengers	20.0	5751	1.41
Skyfall	16.7	5649	4.43
Gravity	16.7	7467	1.29
Frozen	20.8	8084	2.58
The LEGO movie	17.9	5422	3.64
American Sniper	12.1	4152	3.12
Star Wars Episode VII	14.0	4587	6.65
Average	**19.1**	**6457**	**3.4**

- Showing lifetime, total rentals, and chi-squared for each movie fit
- On average, data in blockbuster category is better described by an exponential distribution
- Doesn't seem to be a significant different in lifetime between the categories
- Lifetime is correlated with the year the film was released
  - ➡ More recent films have shorter lifetimes
  - ➡ Could be caused by decreasing library traffic
- Box-office success isn't (positively) correlated with rental numbers
  - ➡ Might be anti-correlated

Conclusion

- Studied movie popularity over time using Seattle Public Library data acquired with MySQL
- Organized the data into histogram form and fit with an exponential distribution
- Was able to address my initial questions
  - How long does interest in a movie last after its release? **About 2 years**
  - Does interest drop-off exponentially? **In some cases, but generally not.**
  - Does box-office success indicate there will be more interest in a movie? **No, in fact it may be the opposite.**
- Possible further studies
  - Expand movie list for each category to get better statistics
  - Explore other functions to fit the data with

Backup

Python code

```python
def GenerateSQL(movies):
  f = open('GetData.sql','w')
  i=1
  for num in movies:
    f.write('SET @bibnum'+str(i)+' = '+num+';\n')
    f.write('SELECT @RelDate'+str(i)+':=MIN(cout) FROM ')
    f.write('spl_2016.outraw WHERE bibNumber = @bibNum'+str(i)+';\n')
    i = i + 1
  f.write('SELECT\n')
  f.write('\tCASE\n')
  i=1
  for num in movies:
    f.write('\t\tWHEN bibNumber = @bibNum'+str(i)+' THEN ')
    f.write('FLOOR(DATEDIFF(cout, @RelDate'+str(i)+')/28)\n')
    i = i + 1
  f.write('\tEND AS MonthsSinceRelease,\n')
  i=1
  for num in movies:
    f.write('COUNT(IF(bibNumber = @bibNum'+str(i)+', ')
    f.write('FLOOR(DATEDIFF(cout, @RelDate'+str(i)+')/28), NULL)) as \''+movies[num]+'\'')
    if i == len(movies):
      f.write('\n')
    else:
      f.write(',\n')
    i = i + 1
  f.write('FROM \n \tspl_2016.outraw\nWHERE\n')
  i=1
  for num in movies:
    f.write('\tbibNumber = @bibNum'+str(i))
    if i == len(movies):
      f.write('\n')
    else:
      f.write(' or \n')
    i = i + 1
  f.write('GROUP BY 1\nORDER BY 1\nLIMIT 24;')
  f.close()
  return
```

- Run using Python v3.7.3
- Takes a dictionary as input
  - Keys: bibNumbers
  - Entries: movie titles
- Generates .sql file

ROOT Code

```
void MakePlots(vector<TString> movie_list) {
  // Generate window to draw on
  TCanvas *can = new TCanvas("can","can",800,800); gStyle->SetOptStat(0);
  TVirtualPad *pad = can->cd(1); pad->SetMargin(0.11,0.05,0.10,0.08);
  for(size_t i = 0; i < movie_list.size(); i++) {
    // Load appropriate row from CSV (%*lg skips a row)
    TString dataStruc = "%lg ";
    for(int j = 0; j < i; j++) dataStruc += "%*lg ";
    dataStruc += "%lg";
    // Load data from file into pairs of (x,y) points
    TGraph *data = new TGraph("Data48.csv",dataStruc," \t,;");
    int nbins = data->GetN();
    // Put data in histogram (for Poisson bin error)
    TH1D *h = new TH1D("h","data hist",nbins,0,nbins);
    for(int ip = 0; ip < nbins; ip++) {
      double x,y;
      data->GetPoint(ip,x,y);
      if(y > 0) {
        h->SetBinContent(ip,y);
        h->SetBinError(ip,sqrt(y));
      }
    }
    // Define function to fit data with
    TF1 *exp = new TF1("exp","[0]*ROOT::Math::exponential_pdf(x,1./[1])",0,nbins);
    exp->SetParameters(300,5);
    // Ensure all parameters are positive
    exp->SetParLimits(0,0,20000); exp->SetParLimits(1,0,50);
    h->Fit(exp);
    // Label everything
    h->SetTitle(movie_list.at(i)+" Activity");
    h->GetXaxis()->SetTitle("Months Since Release"); h->GetYaxis()->SetTitle("Rentals");
    h->Draw("e");
    // Make legend; include fit results
    TLegend *leg = new TLegend(0.59,0.68,0.95,0.9); leg->SetTextSize(.03);
    leg->AddEntry(h,"SPL Data"); leg->AddEntry(exp,"Exponential Fit");
    leg->AddEntry((TObject*)0,TString::Format("#lambda = %.1f (N = %.0f)", exp->GetParameter(1), exp->GetParameter(0)),"");
    leg->AddEntry((TObject*)0,TString::Format("#chi^{2}_{R} = %.2f",exp->GetChisquare()/(nbins-1)),"");
    leg->Draw();
    can->SaveAs("plots/"+to_string(i)+"_"+movie_list.at(i)+".pdf");
  }
  return;
}
```

- Run using the ROOT analysis framework, based in C++
  - https://root.cern.ch
- Takes a list of movie titles as input
- Produces data plots with fits and saves them as PDFs

Potential data quality issues

- Saturation based on rental copies available
  - Tried to use long enough sampling period to mitigate this issue
- Effects of awards on movie popularity
  - Expect movie popularity to decay over time, but may not be the case if awards are given after release
  - In most cases, movies are released to dvd after the Academy Awards, so this shouldn't cause too many issues
- Decrease in library usage over time
  - Hard to quantify the effect of