Advanced models for count data

Frank Edwards

Count data

- Counts are cumulative totals of the number of incidences of some event, generally across time or place

Count data

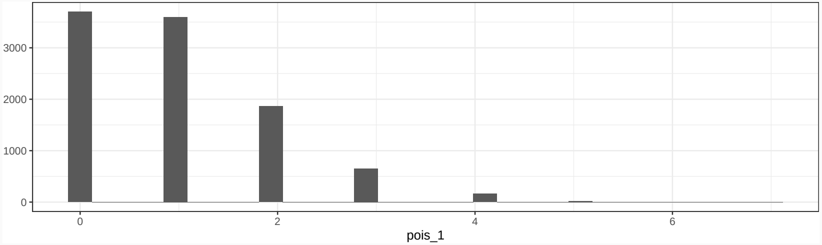
- Counts are cumulative totals of the number of incidences of some event, generally across time or place
- Counts are positive integers $\in [0, \infty]$

Count data

- Counts are cumulative totals of the number of incidences of some event, generally across time or place
- Counts are positive integers $\in [0, \infty]$
- We can model count variables using the Poisson distribution

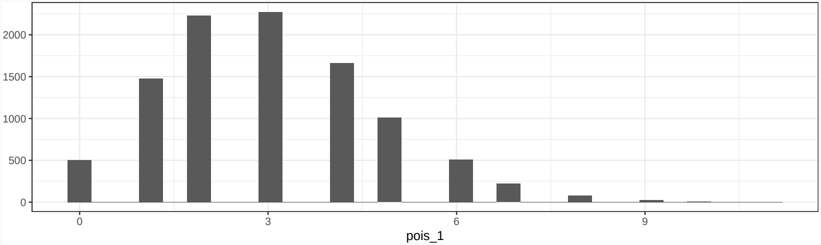
The Poisson distribution (lambda = 1)

```
## pois_1
##    0    1    2    3    4    5    6    7
## 3700 3599 1862  653  161   22    2    1
```



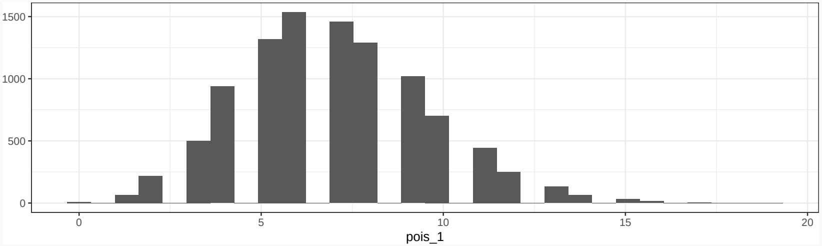
The Poisson distribution (lambda = 3)

```
## pois_1
##    0    1    2    3    4    5    6    7    8    9   10   11
##  505 1476 2231 2273 1662 1009  509  220   80   26    7    2
```

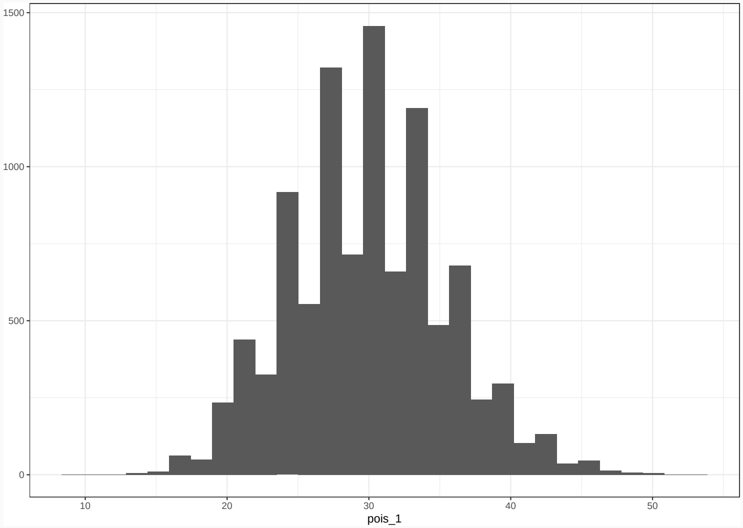


The Poisson distribution (lambda = 7)

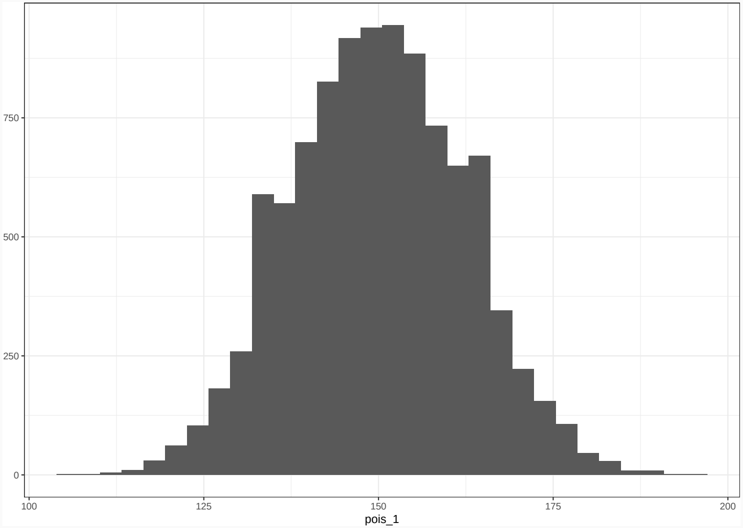
```
## pois_1
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##    9   66  219  500  938 1319 1536 1459 1288 1018  703  444  250  133   63   31
##   16   17   18   19
##   15    6    2    1
```



The Poisson distribution (lambda = 30)



The Poisson distribution (lambda = 150)



Special properties of the Poisson

- The variance and mean of a Poisson variable with parameter λ are both equal to λ

Special properties of the Poisson

- The variance and mean of a Poisson variable with parameter λ are both equal to λ

```
### draw a sample of 10,000 from a Poisson with lambda = 2.3
pois_demo <- rpois(10000, lambda = 2.3)
table(pois_demo)
```

```
## pois_demo
##    0    1    2    3    4    5    6    7    8    9
## 1068 2309 2615 1999 1160  533  219   70   25    2
```

Special properties of the Poisson

- The variance and mean of a Poisson variable with parameter λ are both equal to λ

```
### draw a sample of 10,000 from a Poisson with lambda = 2.3
pois_demo <- rpois(10000, lambda = 2.3)
table(pois_demo)
```

```
## pois_demo
##    0    1    2    3    4    5    6    7    8    9
## 1068 2309 2615 1999 1160  533  219   70   25    2
```

```
mean(pois_demo)
```

```
## [1] 2.2863
```

```
var(pois_demo)
```

```
## [1] 2.345167
```

How well does a Poisson distribution fit our soccer data?

- Is the mean / variance assumption of a Poisson reasonable for our NWSL data on goal scoring?

```r
### Load in NWLS data
library(nwslR)
data(”fieldplayer_overall_season_stats”)
nwsl_stats <- fieldplayer_overall_season_stats
### Check if mean == variance
mean(nwsl_stats$gls)
```

```
## [1] 1.42963
```

```r
var(nwsl_stats$gls)
```

```
## [1] 6.091041
```

Comparing distributions

```r
### Draws from a Poisson with nrow() observations and lambda = mean(nwls$goals)
sim1 <- rpois(nrow(nwsl_stats), lambda = mean(nwsl_stats$gls))
### simulation
table(sim1)
```

```
## sim1
##   0   1   2   3   4   5   6
## 321 461 337 144  67  15   5
```

```r
### observed
table(nwsl_stats$gls)
```

```
## 
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
## 738 237 106  71  61  40  26  16  19  13   5   6   3   3   1   1   2   1   1
```

What’s going on here?

- Problem 1: Player position is associated with goal scoring, right? Defenders don’t score many (or any) goals.

What’s going on here?

- Problem 1: Player position is associated with goal scoring, right? Defenders don’t score many (or any) goals.

What's going on here?

- Problem 1: Player position is associated with goal scoring, right? Defenders don't score many (or any) goals.

```
nwsl_stats %>%
    group_by(pos) %>%
    summarize(gls_mn = mean(gls))
```

```
## # A tibble: 6 x 2
##   pos   gls_mn
##   <chr>  <dbl>
## 1 DF     0.334
## 2 DF,FW  0.925
## 3 DF,MF  0.837
## 4 FW     2.45
## 5 FW,MF  3.17
## 6 MF     1.11
```

So would simulating by position help yield a better fit?

```r
### compute mean, variance, and N for each position
positions <- nwsl_stats %>%
    group_by(pos) %>%
    summarize(obs_mn = mean(gls), obs_var = var(gls), n_obs = n())

positions
```

```
## # A tibble: 6 x 4
##   pos   obs_mn obs_var n_obs
##   <chr>  <dbl>   <dbl> <int>
## 1 DF     0.334   0.558   353
## 2 DF,FW  0.925   2.38     40
## 3 DF,MF  0.837   3.40    129
## 4 FW     2.45    8.63    334
## 5 FW,MF  3.17   14.8     146
## 6 MF     1.11    3.37    348
```

So would simulating by position help yield a better fit?

```r
### now simulate 10000 player - season totals by position
positions <- positions %>%
    group_by(pos) %>%
    mutate(sim_mn = mean(rpois(n_obs, obs_mn)), sim_var = var(rpois(n_obs, obs_mn)))

positions
```

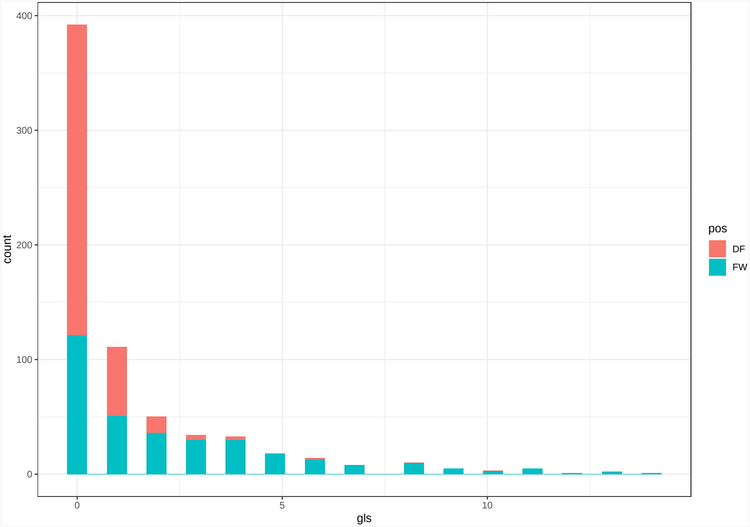
```
## # A tibble: 6 x 6
## # Groups:   pos [6]
##   pos   obs_mn obs_var n_obs sim_mn sim_var
##   <chr>  <dbl>   <dbl> <int>  <dbl>   <dbl>
## 1 DF     0.334   0.558   353  0.323   0.282
## 2 DF,FW  0.925   2.38     40  0.925   0.640
## 3 DF,MF  0.837   3.40    129  0.822   0.785
## 4 FW     2.45    8.63    334  2.18    2.48
## 5 FW,MF  3.17   14.8     146  3.27    3.69
## 6 MF     1.11    3.37    348  1.06    1.28
```

What's going on here?

- Problem 1: Scoring is *clustered* by player position

What’s going on here?

- Problem 1: Scoring is *clustered* by player position



Wait what's a cluster

A cluster is a subset of the population that has similar values to other members of the subset, and systematically different values from other subsets of the population.

Wait what’s a cluster

A cluster is a subset of the population that has similar values to other members of the subset, and systematically different values from other subsets of the population.

Clustering can produce differences in both the expected value and variance of variables across subsets.

Sources of clustering

- A probability distribution assumes that all observations are drawn from the *same* population

Sources of clustering

- A probability distribution assumes that all observations are drawn from the *same* population
- But in practice, most data come from distinct sub-populations (or sub-sub-populations (or sub-sub-sub populations ok you get the joke))

Sources of clustering

- A probability distribution assumes that all observations are drawn from the *same* population
- But in practice, most data come from distinct sub-populations (or sub-sub-populations (or sub-sub-sub populations ok you get the joke))

Sources of clustering in the NWLS data?

Spot the measures that may delineate clusters!

```r
glimpse(nwsl_stats)
```

```
## Rows: 1,350
## Columns: 14
## $ person_id <int> 342, 117, 6, 300, 202, 202, 28, 290, 56, 313, 363, 454, 414,~
## $ season    <dbl> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ~
## $ nation    <chr> "USA", "USA", "ESP", "USA", "USA", "USA", "USA", "USA", "USA~
## $ pos       <chr> "DF", "FW,MF", "FW", "DF,MF", "DF", "DF", "DF", "DF", "MF", ~
## $ team_id   <chr> "WAS", "NJ", "WNY", "KC", "POR", "BOS", "WNY", "SEA", "NJ", ~
## $ mp        <dbl> 5, 20, 15, 22, 4, 11, 7, 22, 7, 20, 3, 2, 1, 22, 20, 6, 1, 7~
## $ starts    <dbl> 4, 20, 14, 22, 2, 11, 2, 22, 5, 11, 0, 1, 0, 22, 16, 5, 0, 4~
## $ min       <dbl> NA, NA, 3, 1900, 212, 990, NA, NA, NA, NA, NA, 123, NA, 1980~
## $ gls       <dbl> 0, 3, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 3, 2, 0, 0, 5, 0, ~
## $ ast       <dbl> 0, 3, NA, 5, 0, 1, NA, 1, 0, 2, 0, 0, 0, 2, 1, 0, 0, 0, 4, 0~
## $ pk        <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ p_katt    <dbl> 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ crd_y     <dbl> 0, 0, 2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, ~
## $ crd_r     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
```

Do we have clusters?

```
positions <- nwsl_stats %>%
    group_by(pos) %>%
    summarize(obs_mn = mean(gls), obs_var = var(gls), n_obs = n())

positions
```

```
## # A tibble: 6 x 4
##   pos   obs_mn obs_var n_obs
##   <chr>  <dbl>   <dbl> <int>
## 1 DF     0.334   0.558   353
## 2 DF,FW  0.925   2.38     40
## 3 DF,MF  0.837   3.40    129
## 4 FW     2.45    8.63    334
## 5 FW,MF  3.17   14.8     146
## 6 MF     1.11    3.37    348
```

But we still have a problem. Does $E(goals|position) = var(goals|position)$?

Overdispersion

Overdispersion is the presence of greater variability than we would expect from a given statistical model

Overdispersion

Overdispersion is the presence of greater variability than we would expect from a given statistical model

For a Poisson model, we assume that

$$var(x) = \bar{x} = \lambda$$

Overdispersion

Overdispersion is the presence of greater variability than we would expect from a given statistical model

For a Poisson model, we assume that

$$var(x) = \bar{x} = \lambda$$

If $var(x) > \bar{x}$, then the data are *overdispersed* relative to predictions from the Poisson model.

Clustering and overdispersion

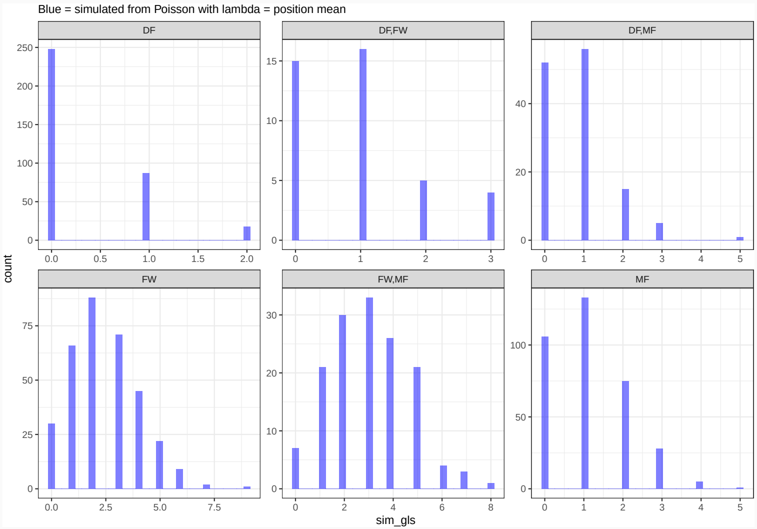
When data come from distinct sub-populations, or clusters, they can have different underlying *data generating processes* (the real world process that produces our observations that we try to approximate using a statistical model).

Clustering and overdispersion

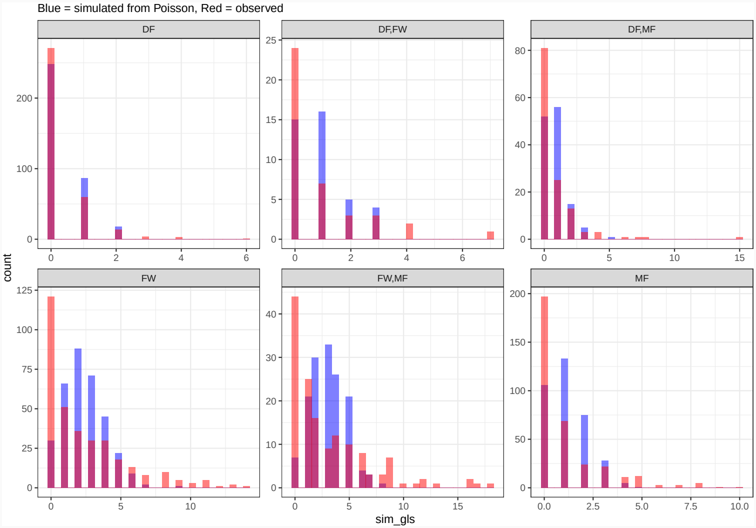
When data come from distinct sub-populations, or clusters, they can have different underlying *data generating processes* (the real world process that produces our observations that we try to approximate using a statistical model).

These differences in data generating processes often result in a) different expected values across sub-groups, and b) different levels of variability across sub-groups

Poisson expectation



Overdispersion



Overdispersion!

```r
nwsl_sim %>%
    group_by(pos) %>%
    summarize(obs_var = var(gls), sim_var = var(sim_gls))
```

```
## # A tibble: 6 x 3
##   pos   obs_var sim_var
##   <chr>   <dbl>   <dbl>
## 1 DF      0.558   0.330
## 2 DF,FW   2.38    0.921
## 3 DF,MF   3.40    0.773
## 4 FW      8.63    2.42
## 5 FW,MF  14.8     2.73
## 6 MF      3.37    1.00
```

Break

Modeling overdispersion: adding a shape parameter

We can relax the $var(x) = \bar{x}$ assumption of the Poisson likelihood with a *quasi-Poisson* likelihood that has the following properties:

$$E(x) = \lambda$$

$$var(x) = \theta\lambda$$

Modeling overdispersion: adding a shape parameter

We can relax the $var(x) = \bar{x}$ assumption of the Poisson likelihood with a *quasi-Poisson* likelihood that has the following properties:

$$E(x) = \lambda$$

$$var(x) = \theta\lambda$$

We call θ an dispersion or shape parameter. Higher values of θ result in more variability, lower values of θ result in more concentration.

The Negative Binomial model

The negative binomial model is very similar to the quasi-poisson. It includes a mean parameter μ and a shape parameter θ.

We can define a negative binomial likelihood as

$$x \sim \text{Negative Binomial}(\mu, \theta)$$

The Negative Binomial model

The negative binomial model is very similar to the quasi-poisson. It includes a mean parameter μ and a shape parameter θ.

We can define a negative binomial likelihood as

$$x \sim \text{Negative Binomial}(\mu, \theta)$$

With an expected value

$$\bar{x} = \mu$$

and variance

$$var(x) = \mu + \frac{\mu^2}{\theta}$$

Let’s see if these likelihoods generate different results

```r
goals_poisson <- glm(gls ~ pos, family = ”poisson”, data = nwsl_stats)

library(MASS)
goals_negbin <- glm.nb(gls ~ pos, data = nwsl_stats)
```

What do we notice about the results?

```
tidy(goals_poisson)
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   -1.10     0.0921    -11.9  1.14e- 32
## 2 posDF,FW       1.02     0.188       5.40 6.59e-  8
## 3 posDF,MF       0.918    0.133       6.89 5.41e- 12
## 4 posFW          1.99     0.0985     20.2  4.56e- 91
## 5 posFW,MF       2.25     0.103      21.8  1.58e-105
## 6 posMF          1.20     0.105      11.4  5.63e- 30
```

```
tidy(goals_negbin)
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -1.10      0.114     -9.58 9.56e-22
## 2 posDF,FW       1.02      0.284      3.58 3.41e- 4
## 3 posDF,MF       0.918     0.187      4.91 9.03e- 7
## 4 posFW          1.99      0.138     14.4  5.67e-47
## 5 posFW,MF       2.25      0.162     13.9  1.20e-43
## 6 posMF          1.20      0.143      8.39 4.85e-17
```

Beta and SE

```
t1 <- tidy(goals_poisson)
t2 <- tidy(goals_negbin)
t1$estimate/t2$estimate
```

```
## (Intercept)    posDF,FW    posDF,MF       posFW    posFW,MF       posMF
##           1           1           1           1           1           1
```

Betas are equal!

Beta and SE

```r
t1 <- tidy(goals_poisson)
t2 <- tidy(goals_negbin)
t1$estimate/t2$estimate
```

```
## (Intercept)    posDF,FW    posDF,MF       posFW    posFW,MF       posMF
##           1           1           1           1           1           1
```

Betas are equal!

```r
t1$std.error/t2$std.error
```

```
## (Intercept)    posDF,FW    posDF,MF       posFW    posFW,MF       posMF
##   0.8049476   0.6630908   0.7124373   0.7112322   0.6350132   0.7376619
```

Standard errors are smaller under Poisson!

Overdispersion and count models

Poisson likelihoods nearly always underestimate standard errors in complex social processes (especially under clustering).

Our models *must* account for overdispersion if we want reasonable uncertainty estimates (standard errors, t-tests, prediction error, etc).

Negative binomial handles this problem well. Other approaches can work too!

Generate predictions

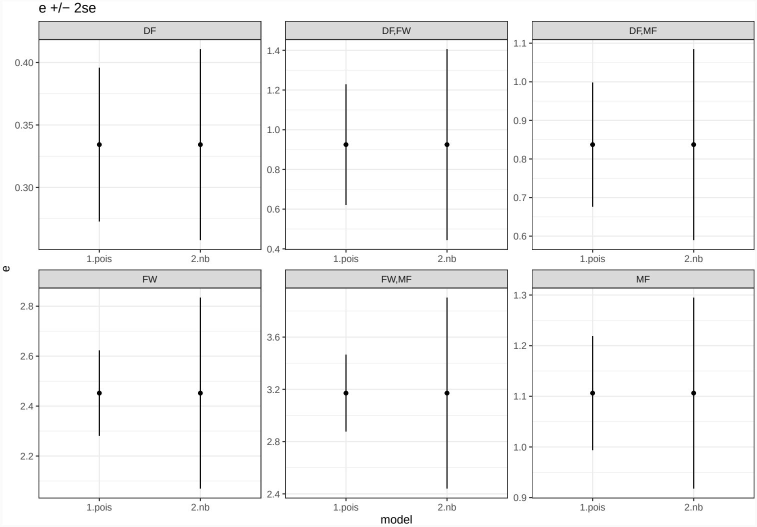
```r
### simulate for each position
fake_data <- data.frame(pos = unique(nwsl_stats$pos))
### from the poisson model
sim_pois <- predict(goals_poisson, type = "response", newdata = fake_data, se.fit = T)
## and the negbin
sim_negbin <- predict(goals_negbin, type = "response", newdata = fake_data, se.fit = T)
```

Format it for plotting

```r
fake_data_pois <- fake_data %>%
    mutate(model = "1.pois", e = sim_pois$fit, se = sim_pois$se.fit)

fake_data_nb <- fake_data %>%
    mutate(model = "2.nb", e = sim_negbin$fit, se = sim_negbin$se.fit)
## glue them together with bind_rows
plot_dat <- bind_rows(fake_data_pois, fake_data_nb)
```

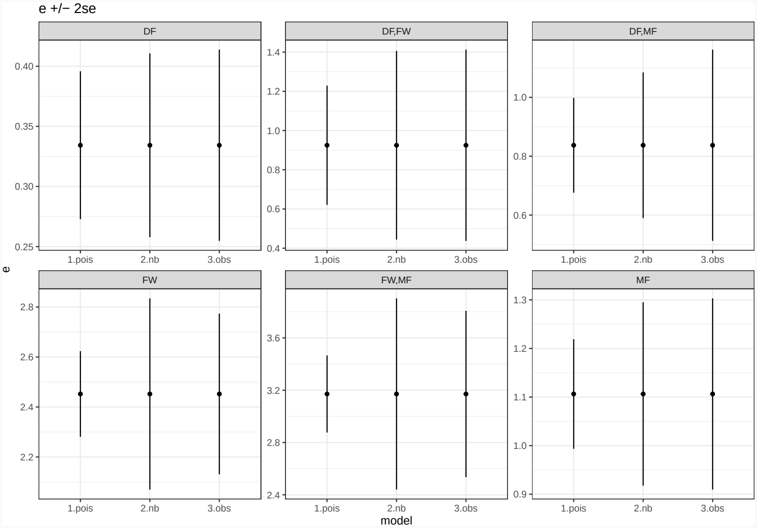
Let's visualize the difference in model predictions



And compare to observed

```
plot_dat <- plot_dat %>%
    bind_rows(nwsl_stats %>%
        group_by(pos) %>%
        summarize(e = mean(gls), se = sd(gls)/sqrt(n())) %>%
        mutate(model = "3.obs"))
```

Visualize



Offsets in event count models

Offsets can improve model fit

Goals are a function of position, sure, but also a function of how many games a player appeared in.

Offsets can improve model fit

Goals are a function of position, sure, but also a function of how many games a player appeared in.

An *offset* term can be added to our model to convert our count into a rate.

Offsets can improve model fit

Goals are a function of position, sure, but also a function of how many games a player appeared in.

An *offset* term can be added to our model to convert our count into a rate.

Here, we can add `mp` as a measure of time

The model for the mean

Using matches played as our offset variable

$$\text{goals} \sim \text{NegBin}(\mu, \theta)$$

$$\log(\mu) = \beta \times \text{position} + \log(\text{mp})$$

A generic form

$$\log(\mu) = \beta X + \log(\text{offset})$$

A generic form

$$\log(\mu) = \beta X + \log(\text{offset})$$

Because $log(x) - log(y) = log(x/y)$ This can be rewritten as

$$\log\left(\frac{\mu}{\text{offset}}\right) = \beta X$$

A generic form

$$\log(\mu) = \beta X + \log(\text{offset})$$

Because $log(x) - log(y) = log(x/y)$ This can be rewritten as

$$\log\left(\frac{\mu}{\text{offset}}\right) = \beta X$$

That's a rate! With the inverse of the link function (*e* here), we can write this as

$$\frac{\mu}{\text{offset}} = e^{\beta X}$$

Let’s fit the model again

```
goals_negbin_offset <- glm.nb(gls ~ pos + offset(log(mp)), data = nwsl_stats)

tidy(goals_negbin_offset)
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   -3.70      0.107    -34.5  9.12e-261
## 2 posDF,FW       1.07      0.255      4.18 2.86e-  5
## 3 posDF,MF       0.761     0.168      4.54 5.59e-  6
## 4 posFW          1.94      0.125     15.5  2.78e- 54
## 5 posFW,MF       2.13      0.141     15.1  1.94e- 51
## 6 posMF          1.16      0.131      8.89 6.17e- 19
```

Now we can compare model fits

```
AIC(goals_negbin, goals_negbin_offset)
```

```
##                     df      AIC
## goals_negbin         7 4011.240
## goals_negbin_offset  7 3627.362
```

The offset dramatically improves our model fit. Let’s see how this works.

The regression parameters

Let’s compute the estimated number of goals under each model for a forward

$$\log(\text{goals}) = \beta_0 + \beta_1 \times \text{position}$$

```r
predict(goals_negbin, newdata = data.frame(pos = "FW"), type = "response")
```

```
##        1
## 2.452096
```

The regression parameters: offset model

Let’s compute the estimated number of goals under each model for a forward

$$\log(\frac{\text{goals}}{\text{games}}) = \beta_0 + \beta_1 \times \text{position}$$

```
predict(goals_negbin_offset, newdata = data.frame(pos = "FW", mp = c(1, 10, 20)),
    type = "response")
```

```
##        1        2        3
## 0.171826 1.718260 3.436521
```

Homework 8 Due in two weeks (4/9)

Replicate (kinda) my paper on police violence, race, and place:
`https://ajph.aphapublications.org/doi/abs/10.2105/AJPH.2018.304559`

1. Load in data on men killed by police by county (./hw/data/fe_division_rural.csv)
2. Compute and visualize death rates for each racial / ethnic group in the data (per 100,000 population)
3. Compute and visualize differences in death rates across Census divisions (`division`) and racial groups
4. Compute and visualize differences in death rates across county types (`ur.code`) and racial groups
5. Estimate a negative binomial model for counts of Black men killed by police using an appropriate offset
6. Estimate a negative binomial model for counts of white men killed by police using an appropriate offset
7. Compare the posterior intervals for expected deaths for Black and white men
8. Estimate new models using predictors for county type and census division and briefly describe your findings
9. Use AIC to compare the models including county and division predictors to the intercept-only models (questions 5 and 6)
10. Explain your findings in 4-8 sentences. Include visuals if helpful.

Data structure for fe_division_rural

Data derived from Fatal Encounters and US Census

- fips: 5 digit county identifier
- state: two letter state abbreviation
- black.men: adult Black male population in county (age >= 18)
- white.men: adult white male population in county
- latino.men: adult Latino population in county
- tot.men: Total adult male population in county
- ur.code: US Dept of Agriculture rural - urban continuum classification for county
- division: US Census geographic division
- d.asian: Asian / PI men killed by police in county (age >= 18)
- d.black: Black men killed by police
- d.latino: Latino men killed by police
- d.other: men with other race/ethnicity identified killed by police
- d.white: white men killed by police
- d.na: men with missing race/ethnicity data killed by policy
- d.total: total men killed by police