Institute
and Faculty
of Actuaries

Probability and Regression in Python and R

Steven Rimmer
Programming for Actuarial Work Working Party

03 October 2018

R input

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 1+2
[1] 3
> rnorm(10,0,1)
 [1]  0.17979525 -1.75712294 -1.12552527  0.03907947  0.04217234
 [6] -0.07170750  0.53504154  0.30017181 -1.11476397  0.65511269
> setwd("C:\\Users\\srimmer")
> source("intro_slide.R")
[1] 9
>
```

https://www.r-project.org/

intro_slide.R - Notepad

File Edit Format View Help

```
a <- 3^2
print(a)
```

- R syntax can by read directly at the command-line interface, or from a separate text file, using `source`
- More sophisticated integrated development environments (IDEs) are also available, such as RStudio.

https://www.rstudio.com/

Institute and Faculty of Actuaries

Quick example: Hypergeometric distribution

```
#-----------------------------------------------------------------------
        white_init      <- 10
        black_init      <- 15
        ndraws          <- 5
        range           <- 0:ndraws
        ns              <- 1e6
        tot_white       <- array(dim=c(ns))
#-----------------------------------------------------------------------
        for (s in 1:ns) {
                white = white_init
                black = black_init
                for (d in 1:ndraws) {
                        if (runif(1, 0, 1) < white/(white+black)) {
                            white = white - 1
                                } else {
                            black = black - 1
                            }
                }
                tot_white[s] = white_init-white
        }
#-----------------------------------------------------------------------
```

- Simulate drawing black and white balls without replacement
- Allocate specific numbers to parameters (`<-`)
- Initialize an `array` with a specific dimension (`dim`)
- Repeat a calculation over an index (`for`)
- Perform a chunk of code when a condition is met (`if…else`)

Institute and Faculty of Actuaries

Quick example: Hypergeometric distribution

```
#-----------------------------------------------------------------------
        library(tibble)
        simulation      <- tibble(
                                sim = 1:ns,
                                count_drawn = tot_white)
#-----------------------------------------------------------------------
```

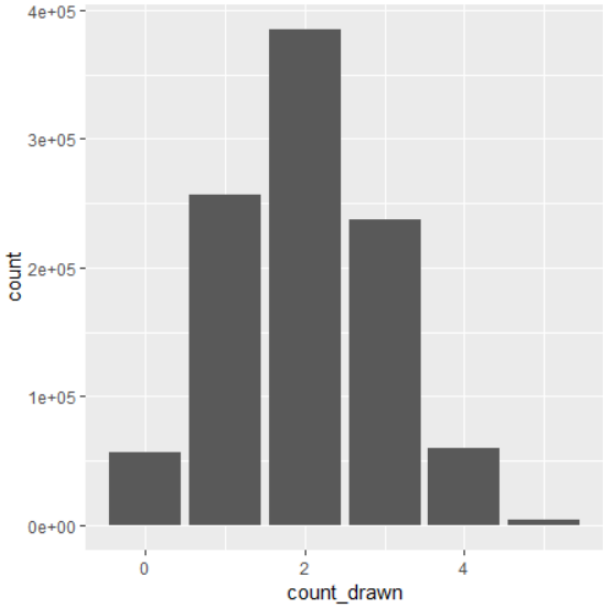
- Call a `library` which helps store information in a table (`tibble`)
- Define a column which labels each simulation (`1:ns`)
- Define a column which stores the output from each simulation (`tot_white`)

```
> simulation
# A tibble: 1,000,000 x 2
     sim count_drawn
   <int>       <dbl>
 1     1           1
 2     2           2
 3     3           3
 4     4           1
 5     5           2
 6     6           3
 7     7           1
 8     8           3
 9     9           2
10    10           1
# ... with 999,990 more rows
```

Institute and Faculty of Actuaries

Quick example: Hypergeometric distribution

```
#-----------------------------------------------------------------------
        library(ggplot2)
        plot    <-  ggplot()+
                        geom_bar(data=simulation,aes(x=count_drawn))
#-----------------------------------------------------------------------
```



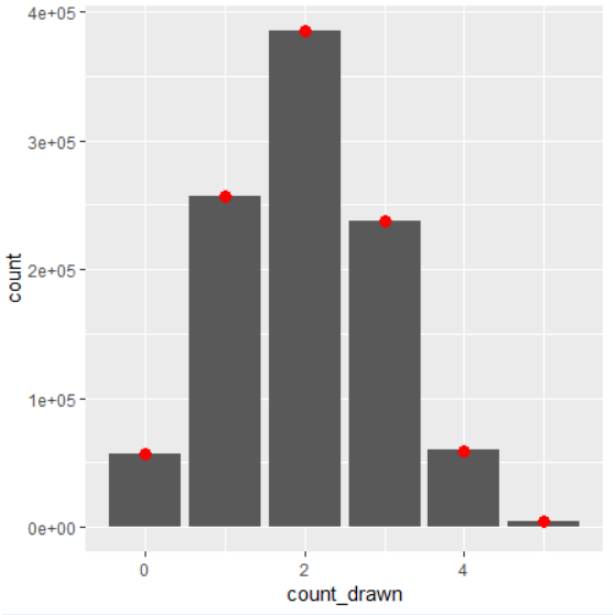
- Call a `library` which supports data visualization (`ggplot2`)
- Make a bar chart (`geom_bar`), plotting the count of white balls drawn on the horizontal axis (`aes(x=…)`)

Institute and Faculty of Actuaries

Quick example: Hypergeometric distribution

```
#-----------------------------------------------------------------------
        check   <- plot + geom_point(
                          aes(x=range,
                               y=ns*dhyper(
                                       x=range,
                                       m=white_init,
                                       n=black_init,
                                       k=ndraws)),
                           colour="red",size=3)
#-----------------------------------------------------------------------
```

- To the original plot, add points (`geom_point`), plotting the theoretical result taken from the in-built hypergeometric distribution (`dhyper`)

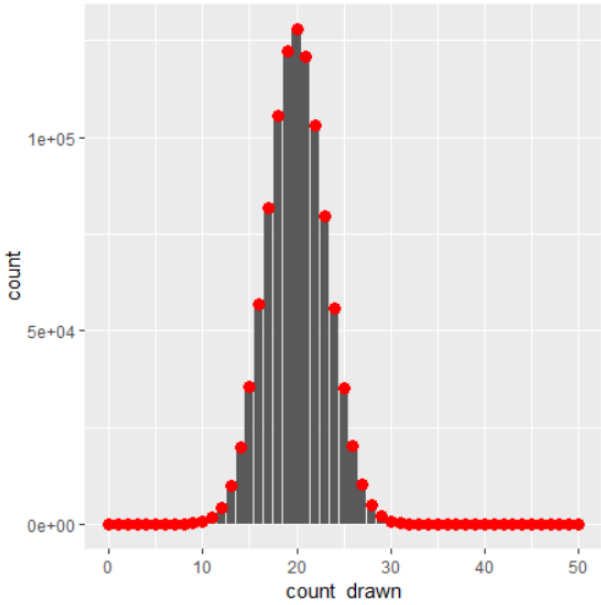


Institute and Faculty of Actuaries

Quick example: Hypergeometric distribution

```
#-----------------------------------------------------------------------
        white_init      <- 100
        black_init      <- 150
        ndraws          <- 50
        range           <- 0:ndraws
        ns              <- 1e6
        tot_white       <- array(dim=c(ns))
#-----------------------------------------------------------------------
```

- Re-run with different parameters



Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        beta0           <- -10
        beta1           <-   0.5
        sigma           <-   8
        n               <-  15
#-----------------------------------------------------------------------
        x               <- sort(runif(n, min=0, 100))
        e               <- rnorm(n, mean = 0, sd = sigma)
        y               <- beta0 + beta1*x + e
#-----------------------------------------------------------------------
```

- Define a linear model: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ where: $\epsilon_i \sim N(0, \sigma^2)$
- Generate `n` values from the uniform distribution (`runif`), order then from low to high (`sort`) and store (`<-`) them as `x`
- Generate `n` values from the normal distribution (`rnorm`) with zero mean and standard deviation, σ, then store them as `e`
- Generate `y` based on the defined model

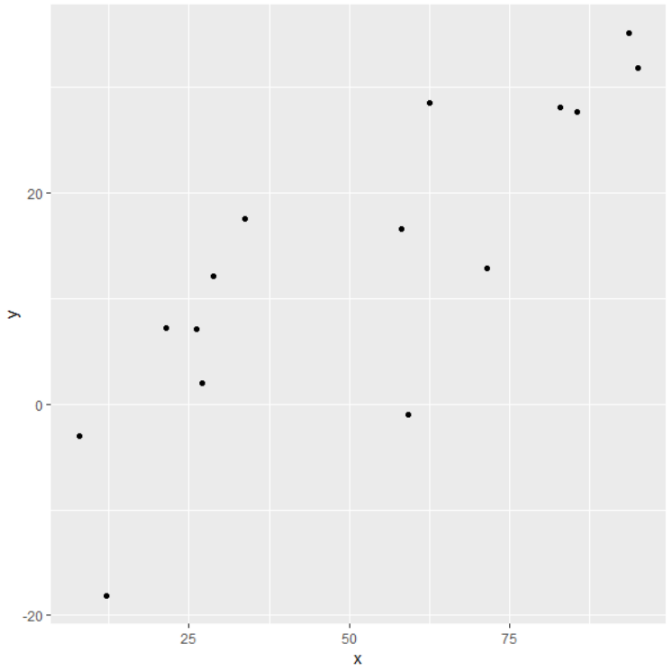
Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        library(tidyverse)
        data_points     <- tibble(sim=1, x=x, y=y)
        plot_first_sim  <- ggplot(data=data_points) +
                                geom_point(mapping = aes(x=x,y=y))
#-----------------------------------------------------------------------
```

- Gather the `n` pairs of `x` and `y` into a table (`tibble`), called `data_points` adding a new variable (`sim`) to label this as the first simulation.
- Plot the data (`ggplot`) as a scatterplot (`geom_point`) showing `x` on the horizontal axis and `y` on the vertical axis.
- The `tibble` and `ggplot` packages are part of a collection called the `tidyverse` , which is called into the before use.

```
> data_points
# A tibble: 15 x 3
     sim     x      y
   <dbl> <dbl>  <dbl>
 1     1  9.96  -8.72
 2     1 12.8  -17.8
 3     1 13.7   -1.38
 4     1 19.5    1.07
 5     1 20.6  -14.3
 6     1 25.4   -8.19
 7     1 27.7   10.6
 8     1 29.5    1.79
 9     1 33.8    5.05
10     1 37.5   11.7
11     1 39.6   26.7
12     1 46.3    6.86
13     1 67.3   25.9
14     1 95.4   34.3
15     1 97.6   48.4
```



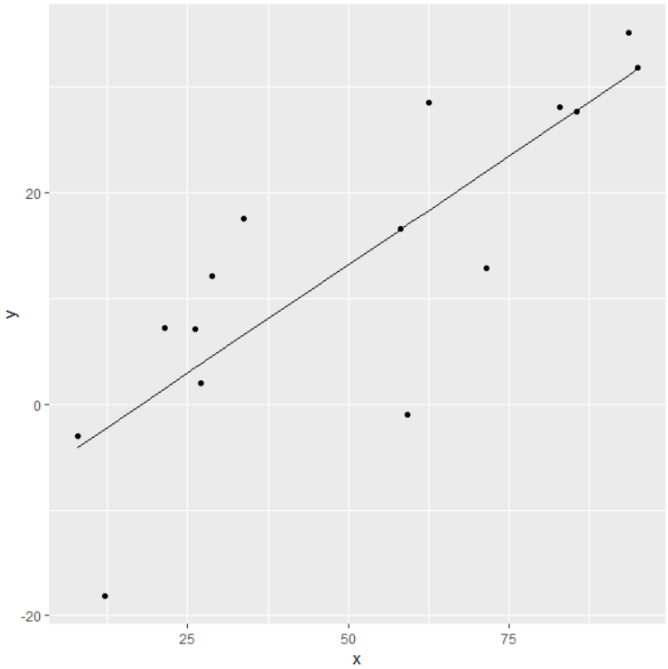
Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        fit <- lm(y~x)
        data_points     <- add_column(data_points,
                                      fitted     = fit$fitted.values
                                      )
        plot_fit        <- ggplot(data=data_points) +
                                geom_point(mapping = aes(x=x,y=y))+
                                geom_line(mapping = aes(x=x,y=fitted))+
                                theme(legend.position = "none")
#-----------------------------------------------------------------------
```

- Fit a linear model (`lm`) with `x` as a linear covariate for the response `y` and store the results in an object called `fit`
- Add a column labelled `fitted` to `data_points`, which stores the resulting values from the linear fit (`fit$fitted.values`)
- Plot the fitted values as a line (`geom_line`) showing `x` on the horizontal axis and `fitted` on the vertical axis.

```
> data_points
# A tibble: 15 x 4
     sim     x       y  fitted
   <dbl> <dbl>   <dbl>   <dbl>
 1     1  9.96  -8.72  -9.20
 2     1 12.8  -17.8   -7.48
 3     1 13.7   -1.38  -6.94
 4     1 19.5    1.07  -3.42
 5     1 20.6  -14.3   -2.70
 6     1 25.4   -8.19   0.191
 7     1 27.7   10.6    1.58
 8     1 29.5    1.79   2.67
 9     1 33.8    5.05   5.32
10     1 37.5   11.7    7.59
11     1 39.6   26.7    8.86
12     1 46.3    6.86  12.9
13     1 67.3   25.9   25.7
14     1 95.4   34.3   42.8
15     1 97.6   48.4   44.1
```



Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

Properties of **`fit`** object

- The results of fitting the linear model were stored in an object called `fit`
- This object has a number of `attributes` which are referenced via the `$` symbol
- On the previous slide the fitted values were added to `data_points` using the `fit$fitted.values` attribute
- Using the `fit$coefficients` attribute, the estimated intercept and slope can be referenced.
- The `fit$residuals` attribute provides the difference between the actual and fitted result.

```
> fit

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -15.2578       0.6085


> attributes(fit)
$`names`
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"


> fit$coefficients
(Intercept)           x
-15.2578233   0.6084606
```

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

Output from **`summary(fit)`**

1. Reports the model which was used
2. Reports the estimated intercept and coefficient of each covariate (the slope)
3. The estimated standard error around that estimate
4. The t-value
5. The corresponding p-value
6. Significant factors highlighted using ‘***’
7. The residual standard error
8. R-squared
9. The F-statistic

```
> summary(fit)

(1) Call:
    lm(formula = y ~ x)

    Residuals:
         Min       1Q   Median       3Q      Max
    -11.5772  -7.2249   0.2478   4.3755  17.8711

    Coefficients:   (2)        (3)       (4)       (5)
                 Estimate Std. Error t value Pr(>|t|)
    (Intercept) -15.25782    3.77101  -4.046  0.00139 ** (6)
    x             0.60846    0.08044   7.564 4.11e-06 ***
    ---
    Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(7)
    Residual standard error: 8.358 on 13 degrees of freedom
(8) Multiple R-squared:  0.8149,    Adjusted R-squared:  0.8006
(9) F-statistic: 57.22 on 1 and 13 DF,  p-value: 4.105e-06
```

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        beta0_hat       <- fit$coefficients[1]
        beta1_hat       <- fit$coefficients[2]
        data_points     <- add_column(data_points,
                                      fitted_check    = beta0_hat+beta1_hat*data_points$x,
                                      true_model      = beta0+beta1*data_points$x
                                      )
#-----------------------------------------------------------------------
```

- Using the `fit$coefficients` attribute, can confirm that the output matches the `fit$fitted.values` attribute
- In addition, the values associated with the true underlying model can be added to the `data_points` table

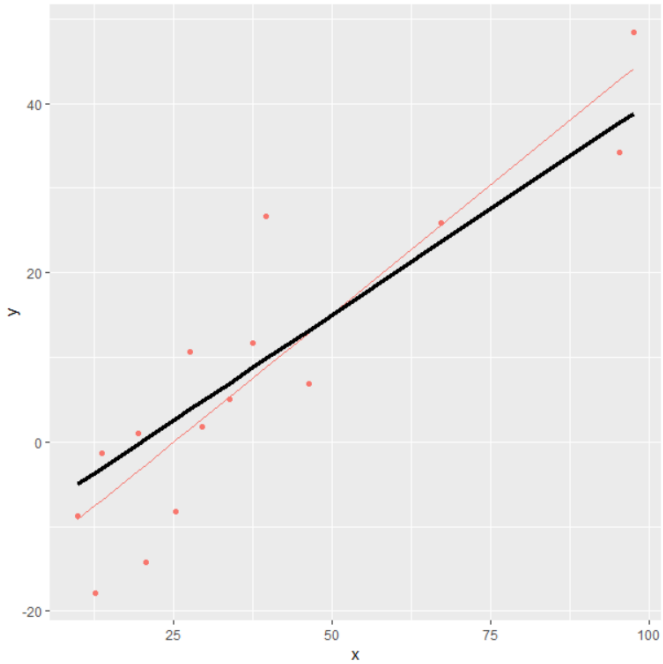
```
> data_points
# A tibble: 15 x 6
     sim     x      y  fitted fitted_check true_model
   <dbl> <dbl>  <dbl>   <dbl>        <dbl>      <dbl>
 1     1  9.96  -8.72  -9.20        -9.20     -5.02
 2     1 12.8  -17.8   -7.48        -7.48     -3.61
 3     1 13.7   -1.38  -6.94        -6.94     -3.17
 4     1 19.5    1.07  -3.42        -3.42     -0.274
 5     1 20.6  -14.3   -2.70        -2.70      0.320
 6     1 25.4   -8.19   0.191        0.191     2.70
 7     1 27.7   10.6    1.58         1.58      3.84
 8     1 29.5    1.79   2.67         2.67      4.73
 9     1 33.8    5.05   5.32         5.32      6.91
10     1 37.5   11.7    7.59         7.59      8.77
11     1 39.6   26.7    8.86         8.86      9.82
12     1 46.3    6.86  12.9         12.9      13.2
13     1 67.3   25.9   25.7         25.7      23.7
14     1 95.4   34.3   42.8         42.8      37.7
15     1 97.6   48.4   44.1         44.1      38.8
```

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        plot_fitvtrue   <- ggplot(data=data_points) +
                                geom_point(mapping = aes(x=x,y=y,colour="red"))+
                                geom_line(mapping = aes(x=x,y=fitted,colour="red"))+
                                geom_line(mapping = aes(x=x,y=true_model),size=1.5)+
                                theme(legend.position = "none")
#-----------------------------------------------------------------------
```

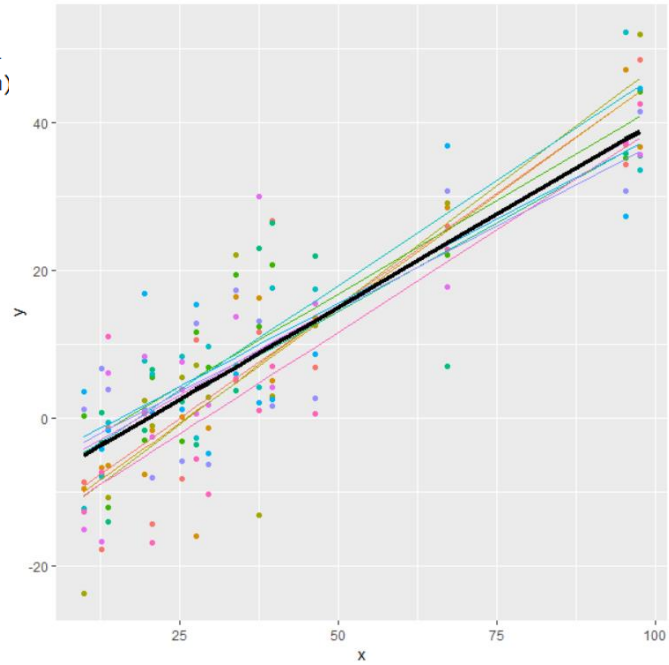
- Plot the true model (in **black**) alongside the fit for this particular realization (in red)



Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        plot_quick_sims <- ggplot(data=data_points) +
                                geom_point(mapping = aes(x=x,y=y,colour=factor(sim)))+
                                geom_line(mapping = aes(x=x,y=fitted,colour=factor(sim)
                                geom_line(mapping = aes(x=x,y=true_model),size=1.5)+
                                theme(legend.position = "none")
#-----------------------------------------------------------------------
```



- Plot the true model (in **black**) alongside the fit for this particular realization (in red)
- Add multiple realizations from the same model (in colours)

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        for (i in 1:ns) {
                e = rnorm(n,mean = 0, sd = sigma)
                y = beta0 + beta1*x + e
                fit = lm(y~x)
                beta0_hat[i]            <- summary(fit)$coefficients[1,1]
                se_beta0_hat[i]         <- summary(fit)$coefficients[1,2]
                beta1_hat[i]            <- summary(fit)$coefficients[2,1]
                se_beta1_hat[i]         <- summary(fit)$coefficients[2,2]
        }
        data_sim <- tibble(sim = 1:ns,
                        beta0_hat,
                        se_beta0_hat,
                        beta1_hat,
                        se_beta1_hat)
#-----------------------------------------------------------------------
```

```
> data_sim
# A tibble: 10,001 x 5
     sim beta0_hat se_beta0_hat beta1_hat se_beta1_hat
   <int>     <dbl>        <dbl>     <dbl>        <dbl>
 1     1    -12.4          1.79     0.560       0.0357
 2     2    -15.0          3.01     0.571       0.0600
 3     3    -10.1          2.65     0.488       0.0528
 4     4    -12.3          3.94     0.472       0.0785
 5     5     -1.24         3.38     0.338       0.0673
 6     6     -5.40         3.58     0.486       0.0712
 7     7    -13.2          4.05     0.542       0.0806
 8     8     -2.76         2.62     0.393       0.0521
 9     9    -13.9          3.92     0.524       0.0781
10    10     -7.09         3.32     0.461       0.0660
# ... with 9,991 more rows
```

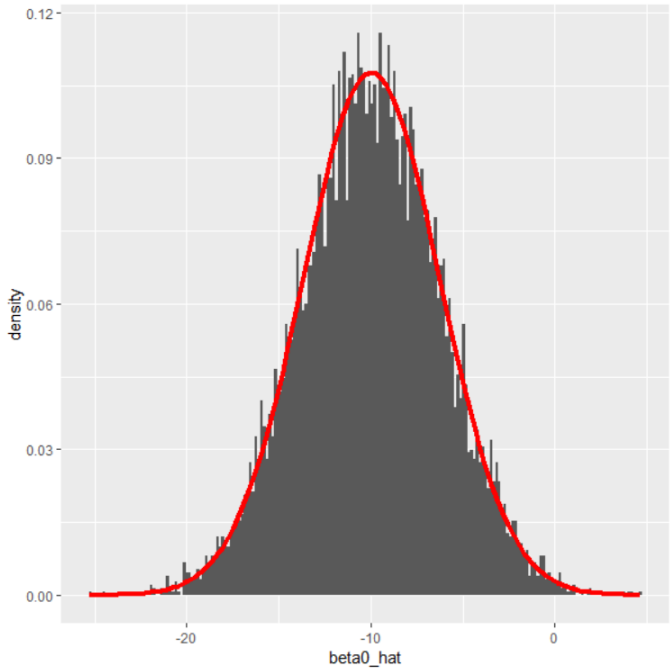
- Iterate a large number (`ns`) of times to generate a distribution of estimates for $\hat{\beta}_0$ (`beta0_hat`) and $\hat{\beta}_1$ (`beta1_hat`)
- Also store the estimate of the associated standard errors for each fit (`se_beta0_hat`, `se_beta1_hat`)

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        sim_beta0_dist <- ggplot(data=data_sim,aes(x=beta0_hat))+
                                geom_histogram(aes(y=..density..),
                                                        bins=200)+
                                stat_function(fun=dnorm,
                                        args=list(
                                                mean=mean(beta0_hat),
                                                sd=mean(se_beta0_hat)),
                                        colour="red",lwd=1.5)
#-----------------------------------------------------------------------
```

- Plot the fitted $\hat{\beta}_0$ (`beta0_hat`) as a histogram (`geom_histogram`), splitting into 200 `bins`
- Add a line (`stat_function`) which shows the normal distribution (`dnorm`) with mean equal to the mean `beta0_hat` and standard deviation equal to the mean `se_beta0_hat`

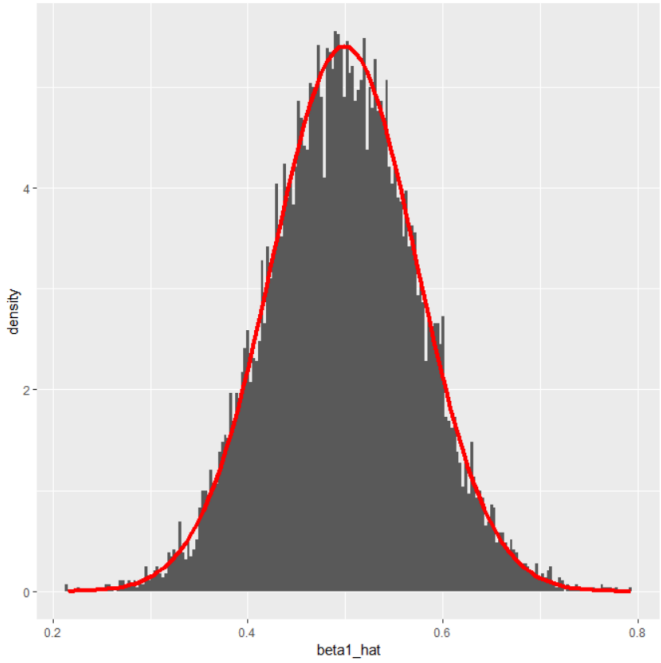


Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        sim_beta1_dist <- ggplot(data=data_sim,aes(x=beta1_hat))+
                                geom_histogram(aes(y=..density..),
                                                        bins=200)+
                                stat_function(fun=dnorm,
                                        args=list(
                                                mean=mean(beta1_hat),
                                                sd=mean(se_beta1_hat)),
                                        colour="red",lwd=1.5)
#-----------------------------------------------------------------------
```

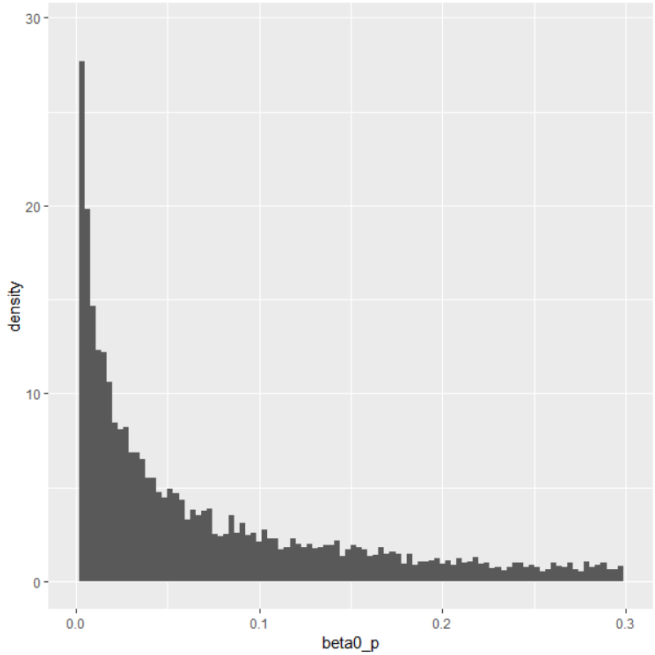
- Plot the fitted $\hat{\beta}_0$ (`beta1_hat`) as a histogram (`geom_histogram`), splitting into 200 `bins`
- Add a line (`stat_function`) which shows the normal distribution (`dnorm`) with mean equal to the mean `beta1_hat` and standard deviation equal to the mean `se_beta1_hat`



Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        for (i in 1:ns) {
                e = rnorm(n,mean = 0, sd = sigma)
                y = beta0 + beta1*x + e
                fit = lm(y~x)
                beta0_t[i]              <- summary(fit)$coefficients[1,3]
                beta0_p[i]              <- summary(fit)$coefficients[1,4]
                beta1_t[i]              <- summary(fit)$coefficients[2,3]
                beta1_p[i]              <- summary(fit)$coefficients[2,4]
        }
        data_tests       <- tibble(sim = 1:ns,
                                  beta0_t
                                  beta0_p,
                                  beta0_t
                                  beta1_p)
        sim_beta0_p_dist <- ggplot(data=data_tests,aes(x=beta0_p))+
                                  geom_histogram(aes(y=..density..),
                                                 bins=100)+
                                  xlim(0,0.3)
#-----------------------------------------------------------------------
```

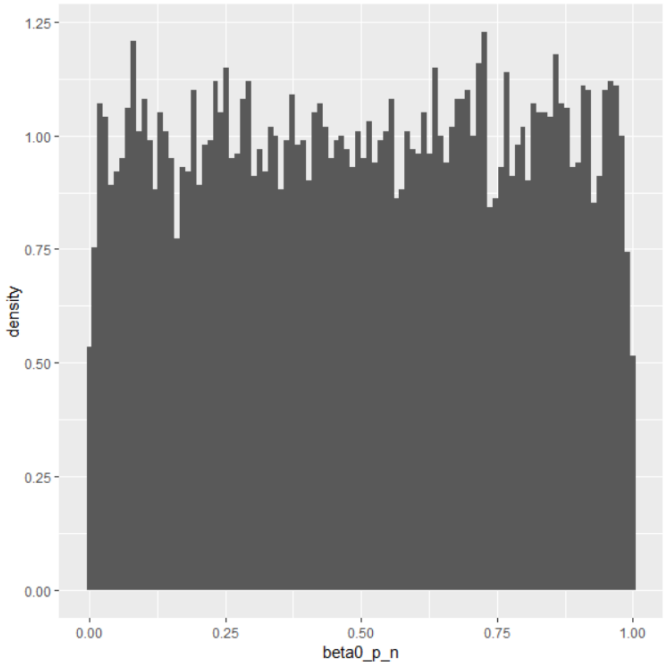


- Store the estimated p-values for each fit (`beta0_p`, `beta1_p`)
- Plot the estimated p-values for $\hat{\beta}_0$ (`beta0_p`) as a histogram (`geom_histogram`), splitting into 100 `bins`, limiting the horizontal axis (`xlim`)

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        for (i in 1:ns) {
                e = rnorm(n,mean = 0, sd = sigma)
                y = beta0 + beta1*x + e
                fit = lm(y~x)
                beta0_t[i]              <- summary(fit)$coefficients[1,3]
                beta0_p[i]              <- summary(fit)$coefficients[1,4]
                beta1_t[i]              <- summary(fit)$coefficients[2,3]
                beta1_p[i]              <- summary(fit)$coefficients[2,4]
        }
        data_tests       <- tibble(sim = 1:ns,
                                  beta0_t
                                  beta0_p,
                                  beta0_t
                                  beta1_p)
        sim_beta0_p_dist <- ggplot(data=data_tests,aes(x=beta0_p))+
                                  geom_histogram(aes(y=..density..),
                                                 bins=100)+
                                  xlim(0,0.3)
#-----------------------------------------------------------------------
```

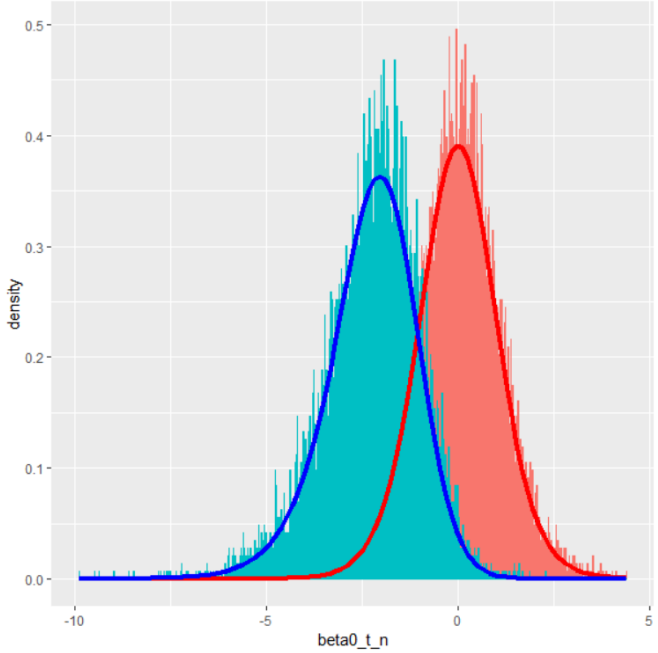


- Generate `ns` simulations of the null hypothesis
- Plot the estimated p-values for $\hat{\beta}_{0,\text{null}}$ (`beta0_p_n`) as a histogram (`geom_histogram`), splitting into 100 `bins`, limiting the horizontal axis (`xlim`)

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        sims_beta0_t_dist        <- ggplot(data=data_tests)+
                                          geom_histogram(aes(x=beta0_t_n,
                                                y=..density..,
                                                alpha=0.5,colour="lightsalmon1"),
                                                bins=1000)+
                                          geom_histogram(aes(x=beta0_t,
                                                y=..density..,
                                                alpha=0.5,colour="skyblue1"),
                                                bins=1000)+
                                          stat_function(fun=dt,
                                                args=list(df=n-2),
                                                colour="red",lwd=1.5)+
                                          stat_function(fun=dt,
                                                args=list(df=n-2,
                                                ncp=mean(beta0_hat)/mean(se_beta0_hat)),
                                                colour="blue",lwd=1.5)+
                                          theme(legend.position = "none")
#-----------------------------------------------------------------------
```



- Plot the estimated t-values for $\hat{\beta}_0$ and $\hat{\beta}_{0,\text{null}}$ (`beta0_t`, `beta0_t_n`) as a histogram (`geom_histogram`) and overlay the t-distribution (`fun=dt`)

Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        sims_beta0_t_dist_data  <- ggplot_build(sims_beta0_t_dist)
        beta0_n_density         <- sims_beta0_t_dist_data$data[[1]]$density*(
                                          sims_beta0_t_dist_data$data[[1]]$xmax-
                                          sims_beta0_t_dist_data$data[[1]]$xmin)
        beta0_density           <- sims_beta0_t_dist_data$data[[2]]$density*(
                                          sims_beta0_t_dist_data$data[[2]]$xmax-
                                          sims_beta0_t_dist_data$data[[2]]$xmin)
#-----------------------------------------------------------------------
```

```
> head(sims_beta0_t_dist_data$data[[1]])
  alpha  colour y count         x      xmin      xmax density
1  0.55 #F8766D 0     0 -14.36850 -14.37818 -14.35882       0
2  0.55 #F8766D 0     0 -14.34914 -14.35882 -14.33946       0
3  0.55 #F8766D 0     0 -14.32977 -14.33946 -14.32009       0
4  0.55 #F8766D 0     0 -14.31041 -14.32009 -14.30073       0
5  0.55 #F8766D 0     0 -14.29104 -14.30073 -14.28136       0
6  0.55 #F8766D 0     0 -14.27168 -14.28136 -14.26200       0
```

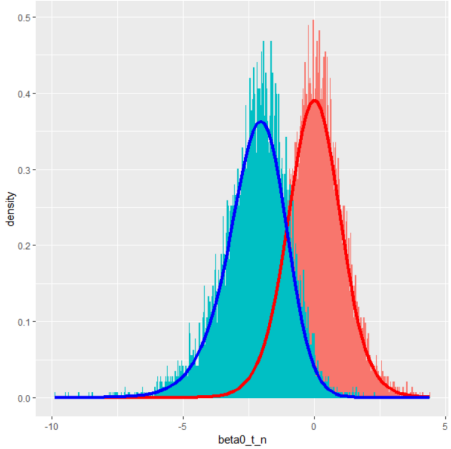
- Using the `ggplot_build` command to values which build up the histogram (or any plot) can be accessed.

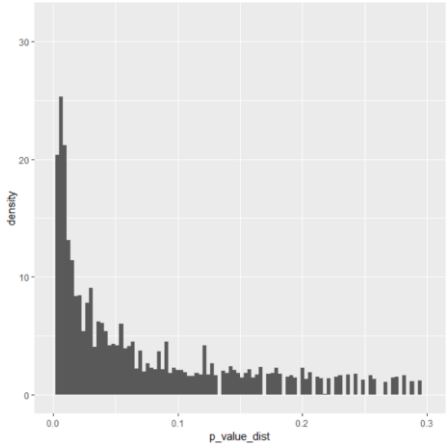
Institute and Faculty of Actuaries

Key statistical tests ‘for free’: Simple linear model

```
#-----------------------------------------------------------------------
        p_value_check <- tibble(beta0_density=beta0_density,
                                extreme=2*pmin(cumsum(beta0_n_density),
                                        rev(cumsum(rev(beta0_n_density)))),
                        )

        p_value_dist <- rep(p_value_check$extreme[1],
                        floor(p_value_check$beta0_density[1]*ns))
        for (i in 2:nrow(p_value_check)) {
                p_value_dist <- c(p_value_dist,rep(p_value_check$extreme[i],
                                        floor(p_value_check$beta0_density[i]*ns)))
                        }
#-----------------------------------------------------------------------
```





- Determining the approximate area under the null distribution curve c (`cumsum`) and the probability density of the observed distribution the distribution of p-values seen before can be estimated.

Institute and Faculty of Actuaries

Wide range of libraries: Generalized linear model

$$\vec{Y} = g^{-1}\left(\underline{X} \cdot \vec{\beta} + \vec{\xi}\right) + \vec{\varepsilon}$$

$$\begin{pmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \end{pmatrix} = \exp\left\{\begin{pmatrix} 1 & 1 & 0 & \dots \\ 1 & 2 & 0 & \dots \\ 1 & 3 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}\begin{pmatrix} \beta_0 \\ \beta_X \\ \beta_A \\ \vdots \end{pmatrix} + \begin{pmatrix} \log(E_1) \\ \log(E_2) \\ \log(E_3) \\ \vdots \end{pmatrix}\right\} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \end{pmatrix}$$

```
#-----------------------------------------------------------------------
        beta0           <- log(0.02)
        betaX           <- 0.015
        betaA           <- 0.08
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
        Data    <- tibble(      X       = rep(1:nX, 4),
                                A       = rep(seq(0, 1), each=2*nX),
                                B       = rep(seq(0, 1), 2, each=nX),
                                ETR     = rep(ETR,4),
                                Eta     = exp(beta0 + betaX*X + betaA*A)
                                )
#-----------------------------------------------------------------------
        Data_sim        <- add_column(Data,
                                      Dth     = rpois(4*nX,Data$ETR*Data$Eta))
#-----------------------------------------------------------------------
```

- Initialize a dummy population, defined by factors, X, A and B.
  - X runs from 1 to 31 (eg an age index)
  - A and B are categorical (eg gender, high pension v low pension)
- Define systematic component to depend on X and A, but not B.
- Simulate poisson events (eg deaths) from the library (`rpois`)

Institute and Faculty of Actuaries

Wide range of libraries: Generalized linear model

```
#-----------------------------------------------------------------------
        Model <- glm(Data_sim$Dth ~     Data_sim$X +
                                        A.F +
                                        B.F +
                                        offset(log(Data_sim$ETR)),
                                        family = poisson(link="log"))
#-----------------------------------------------------------------------
```

- Running the simulation to generate a large number of data points, the estimated $\hat{\beta}$ match the input parameters

```
> summary(Model)

Call:
glm(formula = Data_sim$Dth ~ Data_sim$X + A.F + B.F + offset(log(Data_sim$ETR)),
    family = poisson(link = "log"))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2484  -0.7158   0.1572   0.7198   2.3445

Coefficients:
              Estimate Std. Error   z value Pr(>|z|)
(Intercept) -3.914e+00  1.901e-03 -2058.811   <2e-16 ***
Data_sim$X   1.506e-02  9.415e-05   159.919   <2e-16 ***
A.F1         8.039e-02  1.227e-03    65.531   <2e-16 ***
B.F1         1.407e-03  1.226e-03     1.148    0.251
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 30053.37  on 123  degrees of freedom
Residual deviance:   142.83  on 120  degrees of freedom
AIC: 1587.3

Number of Fisher Scoring iterations: 3
```

Institute and Faculty of Actuaries

Questions

Comments

Institute and Faculty of Actuaries