Examen Final

Yalidt Diaz - 141394
Bruno Gonzalez - 150370

4/12/2019

Final

1. Inferencia gráfica

1. **Preparación de los datos**.

```
wages <- read_csv("data/wages.csv",
                  col_types = cols(
                    id = col_integer(),
                    lnw = col_double(),
                    exper = col_double(),
                    ged = col_double(),
                    postexp = col_double(),
                    black = col_integer(),
                    hispanic = col_integer(),
                    hgc = col_integer(),
                    hgc.9 = col_integer()
                  ))
```

- Selecciona los sujetos con grado de estudios completado igual a 9, 10 u 11.

```
tidy_wages <- wages %>%
  filter(hgc >= 9, hgc <=11) %>%
  dplyr::select(1:8)
```

- Elimina las observaciones donde el logaritmo del salario (lnw) es mayor a 3.5.

```
tidy_wages <- tidy_wages %>%
  filter(lnw <= 3.5)
```

- Crea una variable correspondiente a raza, un sujeto es de raza hispana si la variable hispanic toma el valor 1, de raza negra si la variable black toma el valor 1 y de raza blanca si las dos anteriores son cero.

```
tidy_wages <- tidy_wages %>%
  mutate(raza = ifelse(hispanic==1,'hispana',ifelse(black==1,'negra','blanca')))
```

- Crea un subconjunto de la base de datos de tal manera que tengas el mismo número de sujetos distintos en cada grupo de raza. Nota: habrá el mismo número de sujetos en cada grupo pero el número de observaciones puede diferir pues los sujetos fueron visitados un número distinto de veces.

Primero creamos una función que tomo como variable el número de sujetos que se desean incluir

```
f_wages_sample <- function(n=10){
  sample <- tidy_wages %>%
    group_by(id, raza) %>%
    summarize() %>%
    ungroup() %>%
    group_by(raza) %>%
    sample_n(n)

  tidy_wages %>%
```

```
    semi_join(sample)
}
```

Posteriormente creamos el subconjunto de datos.

```
sample_wages <- f_wages_sample(80)
```

2 Prueba de hipótesis visual

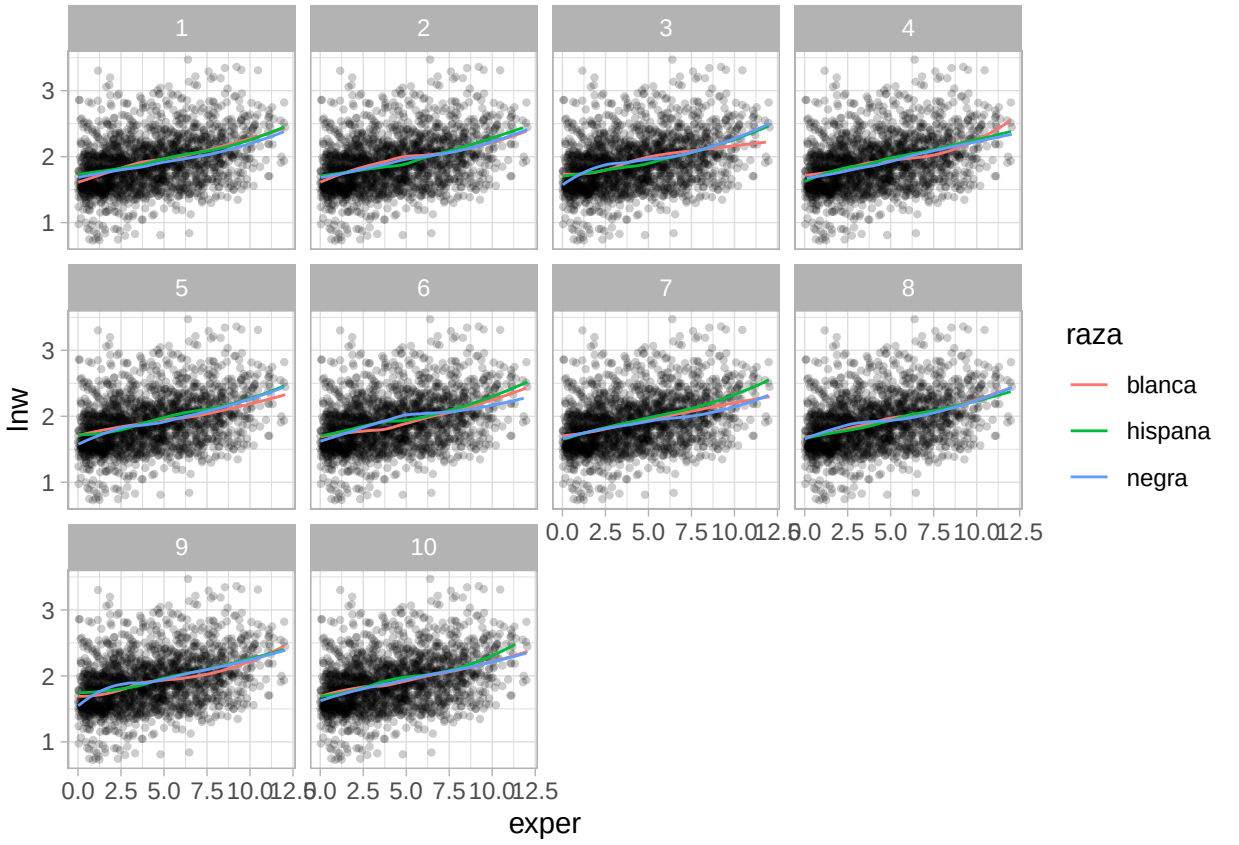
- El escenario nulo consiste en que no hay diferencia entre las razas. Para generar los datos nulos, la etiqueta de raza de cada sujeto se permuta, es decir, se reasigna la raza de cada sujeto de manera aleatoria (para todas las mediciones de un sujeto dado se reasigna una misma raza). Genera 9 conjuntos de datos nulos y para cada uno ajusta una curva *loess* siguiendo la instrucción de la gráfica de arriba. Crea una gráfica de paneles donde incluyas los 9 conjuntos nulos y los datos reales, estos últimos estarán escondidos de manera aleatoria.

Primero creamos los datos nulos

```
null_wages <- lineup(null_permute('raza'),
                     n = 10,
                     sample_wages)
```

A continuación hacemos las gráficas correspondientes

```
ggplot(null_wages, aes(x = exper, y = lnw)) +
  geom_point(alpha = 0.2, size = 0.8) +
  geom_smooth(aes(group = raza, color = raza), method = "loess", se = FALSE, size=0.5) +
  facet_wrap(~.sample) +
  theme_light()
```



- Realiza la siguiente pregunta a una o más personas **que no tomen la clase**:

_Las siguientes 10 gráficas muestran suavizamientos de log(salarios) por años de experiencia laboral. Una de ellas usa datos reales y las otras 9 son datos nulos, generados bajo el supuesto de que no existe diferencia entre los subgrupos. ¿Cuál es la gráfica más distinta?

Reporta si las personas cuestionadas pudieron distinguir los datos. Se le preguntó a 5 personas, de las cuales solo 2 pudieron distinguir los datos correctos.

- ¿Cuál es tu conclusión de la prueba de hipótesis visual? No se puede rechazar la hipotesis nula, por lo que no se puede deducir que existe diferencia en los salarios con base en los años trabajados entre los subgrupos.
- ¿A cuántas personas preguntaste y cuál es el valor p de la prueba? Dado que solo 2 personas de 5 distinguieron la gráfica con los datos reales (rechazaron la hipótesis nula), el vaolor p es de 0.6.

2. Simulación para el cálculo de tamaños de muestra

Utilizarás simulación y los resultados de las elecciones de gobernador en Guanajuato correspondientes al 2012. En el caso de **MAS**, para cada tamaño de muestra $n = 50, 100, 200, 300, 400, 500, 600, 700$:

i. Simula una muestra aleatoria de tamaño n.

```
gto_2012 <- read.csv("./data/gto_2012.csv")

muestra <- map(c(50,100,200,300,400,500,600,700), ~sample_n(gto_2012,.))
glimpse(muestra[[1]])
```

```
## Observations: 50
## Variables: 23
## $ casilla_id      <int> 289, 6441, 2339, 745, 5295, 6547, 2594, 5306, ...
## $ distrito_fed_17 <int> 1, 10, 4, 3, 14, 14, 2, 14, 14, 1, 10, 8, 13, ...
## $ distrito_fed_12 <int> 1, 10, 4, 5, 10, 14, 2, 10, 14, 1, 13, 8, 13, ...
## $ distrito_loc_17 <int> 2, 20, 8, 3, 15, 22, 9, 15, 22, 2, 19, 14, 19,...
## $ distrito_loc_12 <int> 2, 21, 8, 3, 17, 22, 9, 17, 22, 2, 21, 13, 19,...
## $ are             <fct> 1-2, 10-43, 4-48, 5-17, 10-52, 14-20, 2-14, 10...
## $ seccion         <dbl> 332, 2340, 891, 1708, 2764, 97, 169, 2771, 62,...
## $ casilla         <fct> B-C, B-C, B-C, B-C, B-C, B-C, B-C, B-C, B-C, B...
## $ tipo_seccion    <fct> R, R, R, U, R, U, U, U, M, M, U, U, R, R, M, R...
## $ pri_pvem        <int> 75, 37, 62, 138, 85, 62, 149, 67, 88, 158, 110...
## $ pan_na          <int> 79, 123, 94, 246, 114, 75, 210, 272, 151, 143,...
## $ prd             <int> 0, 12, 21, 12, 12, 46, 22, 12, 34, 21, 18, 13,...
## $ pt              <int> 0, 1, 5, 6, 1, 3, 6, 4, 8, 14, 2, 8, 7, 2, 3, ...
## $ mc              <int> 0, 0, 0, 7, 0, 0, 3, 2, 3, 2, 2, 5, 1, 2, 2, 1...
## $ otros           <int> 1, 0, 6, 41, 10, 16, 17, 25, 13, 22, 14, 7, 12...
## $ total           <int> 155, 173, 188, 450, 222, 202, 407, 382, 297, 3...
## $ ln              <int> 181, 397, 301, 593, 525, 487, 675, 607, 475, 5...
## $ tamano_md       <int> 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1...
## $ tamano_gd       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ region          <int> 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2...
## $ casilla_ex      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ rural           <int> 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1...
## $ ln_total        <int> 181, 397, 301, 593, 525, 487, 675, 607, 475, 5...
```

La lista muestra contiene todas las muestras, solo se mustra el resumen de la primera.

ii. Calcula el estimador de razón (correspondiente a muestreo aleatorio simple) para cada candidato:

$$\hat{p} = \frac{\sum_i Y_i}{\sum_i X_i}$$

$$\hat{p} = \frac{\sum_h \frac{N_h}{n_h} \sum_i Y_{hi}}{\sum_h \frac{N_h}{n_h} \sum_i X_{hi}}$$

Primero definimos una función para calcular la muestra

```
estimador<-function(data,size){
  data %>%
    sample_n(size) %>%
    mutate(muestra=glue("muestra_{size}")) %>%
    summarise(p_pri=sum(pri_pvem)/sum(total),p_pan=sum(pan_na)/sum(total),
              p_prd=sum(prd)/sum(total),p_pt=sum(pt)/sum(total),
              p_mc=sum(mc)/sum(total),p_otros=sum(otros)/sum(total)) %>%
    select(p_pri,p_pan,p_prd,p_pt,p_mc,p_otros)
}
```

Con esta función podemos calcular los estimadores para la muestra que se desee.

```
estimador(gto_2012, 50)
```

```
##       p_pri     p_pan      p_prd       p_pt        p_mc    p_otros
## 1 0.4125928 0.4539947 0.06046926 0.01378081 0.006712207 0.05245025
```

iii. Repite los pasos i y ii 1000 veces para estimar el error estándar para una muestra de tamaño n.

```
m_50<-rerun(1000,estimador(gto_2012,50)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
                                                              se_mc=sd(p_mc),
                                                              se_otros=sd(p_otros))
m_100<-rerun(1000,estimador(gto_2012,100)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
                                                               se_mc=sd(p_mc),
                                                               se_otros=sd(p_otros))
m_200<-rerun(1000,estimador(gto_2012,200)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

m_300<-rerun(1000,estimador(gto_2012,300)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
                                                               se_mc=sd(p_mc),
                                                               se_otros=sd(p_otros))
m_400<-rerun(1000,estimador(gto_2012,400)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
                                                              se_mc=sd(p_mc),
                                                              se_otros=sd(p_otros))
m_500<-rerun(1000,estimador(gto_2012,500)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
```

```
                                                                    se_mc=sd(p_mc),
                                                                    se_otros=sd(p_otros))
m_600<-rerun(1000,estimador(gto_2012,600)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
                                                                    se_mc=sd(p_mc),
                                                                    se_otros=sd(p_otros))
m_700<-rerun(1000,estimador(gto_2012,700)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
                                                                    se_mc=sd(p_mc),
                                                                    se_otros=sd(p_otros))
```

Así, el error estadar para cada una de las muestras será:

```
MAS_se <- rbind(m_50,m_100,m_200,m_300,m_400,m_500,m_600,m_700)
rownames(MAS_se) <- c('50','100','200','300','400','500','600','700')
MAS_se
```

```
##          se_pri      se_pan      se_prd        se_pt       se_mc
## 50  0.015121475 0.013595327 0.007645076 0.0020649774 0.003810013
## 100 0.010681863 0.009718143 0.005546981 0.0014943659 0.002635907
## 200 0.007543489 0.006820766 0.004010425 0.0010280244 0.001986986
## 300 0.006196002 0.005797762 0.003154756 0.0008506467 0.001505506
## 400 0.005117751 0.004787805 0.002658726 0.0007149066 0.001337012
## 500 0.004386163 0.004107723 0.002411291 0.0006289142 0.001139603
## 600 0.004177852 0.003948906 0.002222419 0.0006167743 0.001031706
## 700 0.003850842 0.003542728 0.002061846 0.0005406360 0.000974644
##        se_otros
## 50  0.004336405
## 100 0.002844230
## 200 0.002167815
## 300 0.001698118
## 400 0.001463433
## 500 0.001203336
## 600 0.001141289
## 700 0.001051971
```

Para cada **estratificación** (`distrito_fed_17` y `distrito_loc_17`) y tamaño de muestra $n = 50, 100, 200, 300, 400, 500, 600, 700$:

i. Simula una muestra estratificada de tamaño n, donde el tamaño de muestra en cada estrato se asigna proporcional al tamaño del estrato, esto es, sea N_h el número de casillas en el h-ésimo estrato, entonces para el estrato h el número de casillas en la muestra será:

$$n_h = N_h \cdot \frac{n}{\sum_j N_j}$$

ii. Calcula el estimador de razón combinado (correspondiente a muestreo estratificado) para cada candidato:

$$\hat{p} = \frac{\sum_h \frac{N_h}{n_h} \sum_i Y_{hi}}{\sum_h \frac{N_h}{n_h} \sum_i X_{hi}}$$

iii. Repite los pasos i y ii 1000 veces para estimar el error estándar para una muestra de tamaño n.

Agrupamos el dataset por los estratos solicitados

```
####### muestra por estratos ########
by_stratum_dist_17<-gto_2012 %>%
  group_by(distrito_fed_17) %>%
  arrange(distrito_fed_17)

by_stratum_local_17<-gto_2012 %>%
  group_by(distrito_loc_17) %>%
  arrange(distrito_loc_17)
```

Encontramos la n proporcional a cada estrato con las diferentes muestras.

```
#Para el estrato distrito fed 17:
size_estrato_fed_17<-ssampcalc(df = gto_2012,n =50,strata = distrito_fed_17) %>%
  cbind(ssampcalc(df = gto_2012,n =100,strata = distrito_fed_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n =200,strata = distrito_fed_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n =300,strata = distrito_fed_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n =400,strata = distrito_fed_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n =500,strata = distrito_fed_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n =600,strata = distrito_fed_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n =700,strata = distrito_fed_17)[,4])

names(size_estrato_fed_17)<-c("distrito_fed_17","Nh","wt","nh_50","nh_100","nh_200","nh_300","nh_400",
         "nh_500","nh_600","nh_700")

size_estrato_fed_17
```

```
##    distrito_fed_17  Nh          wt nh_50 nh_100 nh_200 nh_300 nh_400
## 1                1 421 0.062407353     3      6     12     19     25
## 2                2 463 0.068633264     3      7     14     21     27
## 3                3 433 0.064186184     3      6     13     19     26
## 4                4 421 0.062407353     3      6     12     19     25
## 5                5 443 0.065668544     3      7     13     20     26
## 6                6 332 0.049214349     2      5     10     15     20
## 7                7 494 0.073228580     4      7     15     22     29
## 8                8 497 0.073673288     4      7     15     22     29
## 9                9 388 0.057515565     3      6     12     17     23
## 10              10 539 0.079899200     4      8     16     24     32
## 11              11 343 0.050844945     3      5     10     15     20
## 12              12 479 0.071005040     4      7     14     21     28
## 13              13 518 0.076786244     4      8     15     23     31
## 14              14 536 0.079454492     4      8     16     24     32
## 15              15 417 0.061814409     3      6     12     19     25
## 16              20  22 0.003261192     0      0      1      1      1
##    nh_500 nh_600 nh_700
## 1      31     37     44
## 2      34     41     48
## 3      32     39     45
## 4      31     37     44
## 5      33     39     46
## 6      25     30     34
## 7      37     44     51
## 8      37     44     52
## 9      29     35     40
## 10     40     48     56
## 11     25     31     36
```

```
## 12     36     43     50
## 13     38     46     54
## 14     40     48     56
## 15     31     37     43
## 16      2      2      2
```

```
#Para el estrato distrito loc 17:
size_estrato_loc_17<-ssampcalc(df = gto_2012,n = 50,strata = distrito_loc_17) %>%
  cbind(ssampcalc(df = gto_2012,n = 100,strata = distrito_loc_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n = 200,strata = distrito_loc_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n = 300,strata = distrito_loc_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n = 400,strata = distrito_loc_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n = 500,strata = distrito_loc_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n = 600,strata = distrito_loc_17)[,4]) %>%
  cbind(ssampcalc(df = gto_2012,n = 700,strata = distrito_loc_17)[,4])

names(size_estrato_loc_17)<-c("distrito_loc_17","Nh","wt","nh_50","nh_100","nh_200","nh_300","nh_400",
                              "nh_500","nh_600","nh_700")

size_estrato_loc_17
```

```
##    distrito_loc_17  Nh         wt nh_50 nh_100 nh_200 nh_300 nh_400 nh_500
## 1                1 303 0.04491551     2      4      9     13     18     22
## 2                2 330 0.04891788     2      5     10     15     20     24
## 3                3 248 0.03676253     2      4      7     11     15     18
## 4                4 218 0.03231545     2      3      6     10     13     16
## 5                5 256 0.03794841     2      4      8     11     15     19
## 6                6 279 0.04135784     2      4      8     12     17     21
## 7                7 369 0.05469908     3      5     11     16     22     27
## 8                8 295 0.04372962     2      4      9     13     17     22
## 9                9 303 0.04491551     2      4      9     13     18     22
## 10              10 314 0.04654610     2      5      9     14     19     23
## 11              11 291 0.04313667     2      4      9     13     17     22
## 12              12 316 0.04684257     2      5      9     14     19     23
## 13              13 278 0.04120961     2      4      8     12     16     21
## 14              14 336 0.04980729     2      5     10     15     20     25
## 15              15 346 0.05128965     3      5     10     15     21     26
## 16              16 271 0.04017195     2      4      8     12     16     20
## 17              17 255 0.03780018     2      4      8     11     15     19
## 18              18 371 0.05499555     3      5     11     16     22     27
## 19              19 367 0.05440261     3      5     11     16     22     27
## 20              20 419 0.06211088     3      6     12     19     25     31
## 21              21 203 0.03009191     2      3      6      9     12     15
## 22              22 378 0.05603320     3      6     11     17     22     28
##    nh_600 nh_700
## 1      27     31
## 2      29     34
## 3      22     26
## 4      19     23
## 5      23     27
## 6      25     29
## 7      33     38
## 8      26     31
## 9      27     31
## 10     28     33
```

```
## 11     26     30
## 12     28     33
## 13     25     29
## 14     30     35
## 15     31     36
## 16     24     28
## 17     23     26
## 18     33     38
## 19     33     38
## 20     37     43
## 21     18     21
## 22     34     39
```

Función Muestra simulada proporcional a los estratos

```
set.seed(141394)
estimador_strata<-function(stratum,nh,data){
  index_<-stratsrs(stratum,nh)
  data[index_,] %>%
    summarise(p_pri=sum(pri_pvem)/sum(total),p_pan=sum(pan_na)/sum(total),
                              p_prd=sum(prd)/sum(total),p_pt=sum(pt)/sum(total),
                              p_mc=sum(mc)/sum(total),p_otros=sum(otros)/sum(total)) %>%
    select(p_pri,p_pan,p_prd,p_pt,p_mc,p_otros)
}
```

```
#Estrado distrito federal 17
st1_50<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
                                    size_estrato_fed_17$nh_50,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
                        se_mc=sd(p_mc),
                        se_otros=sd(p_otros))

st1_100<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
                                    size_estrato_fed_17$nh_100,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st1_200<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
                                    size_estrato_fed_17$nh_200,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st1_300<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
                                    size_estrato_fed_17$nh_300,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st1_400<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
```

```
                                    size_estrato_fed_17$nh_400,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st1_500<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
                                    size_estrato_fed_17$nh_500,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st1_600<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
                                    size_estrato_fed_17$nh_600,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st1_700<-rerun(1000,estimador_strata(by_stratum_dist_17$distrito_fed_17,
                                     size_estrato_fed_17$nh_700,by_stratum_dist_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))
```

```
#Estrado distrito local 17

st2_50<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                    size_estrato_loc_17$nh_50,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st2_100<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                    size_estrato_loc_17$nh_100,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st2_200<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                    size_estrato_loc_17$nh_200,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st2_300<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                    size_estrato_loc_17$nh_300,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
```

```
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st2_400<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                     size_estrato_loc_17$nh_400,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st2_500<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                     size_estrato_loc_17$nh_500,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st2_600<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                     size_estrato_loc_17$nh_600,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))

st2_700<-rerun(1000,estimador_strata(by_stratum_local_17$distrito_loc_17,
                                     size_estrato_loc_17$nh_700,by_stratum_local_17)) %>%
  bind_rows() %>%
  summarise(se_pri=sd(p_pri),se_pan=sd(p_pan),se_prd=sd(p_prd),se_pt=sd(p_pt),
            se_mc=sd(p_mc),
            se_otros=sd(p_otros))
```

Ahora:

1. Reporta en una tabla el error estándar para cada candidato, tamaño de muestra y diseño (MAS y las dos estratificaciones propuestas).

```
tablaSe<-m_50 %>%
  rbind(m_100,m_200,m_300,m_400,m_500,m_600,m_700,st1_50,st1_100,st1_200,st1_300,st1_400,st1_500,st1_60
        st2_50,st2_100,st2_200,st2_300,st2_400,st2_500,st2_600,st2_700)
tablaSe$sample <-c("MAS","MAS","MAS","MAS","MAS","MAS","MAS","MAS",
                   "1ST","1ST","1ST","1ST","1ST","1ST","1ST","1ST",
                   "2ST","2ST","2ST","2ST","2ST","2ST","2ST","2ST")
tablaSe$n<-c(50,100,200,300,400,500,600,700,50,100,200,300,400,500,600,700,50,100,200,300,400,500,600,7

tablaSe
```

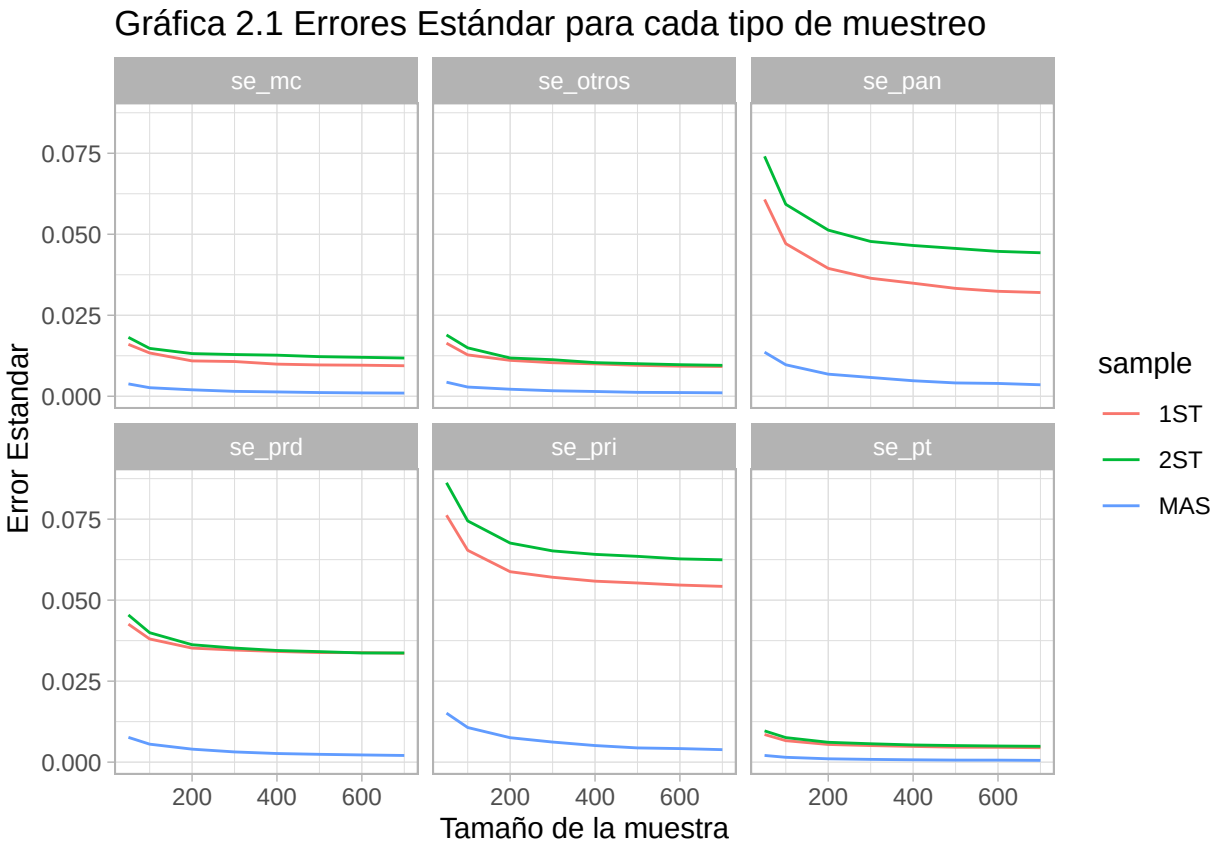
```
##         se_pri      se_pan      se_prd        se_pt       se_mc
## 1  0.015121475 0.013595327 0.007645076 0.0020649774 0.003810013
## 2  0.010681863 0.009718143 0.005546981 0.0014943659 0.002635907
## 3  0.007543489 0.006820766 0.004010425 0.0010280244 0.001986986
## 4  0.006196002 0.005797762 0.003154756 0.0008506467 0.001505506
## 5  0.005117751 0.004787805 0.002658726 0.0007149066 0.001337012
## 6  0.004386163 0.004107723 0.002411291 0.0006289142 0.001139603
## 7  0.004177852 0.003948906 0.002222419 0.0006167743 0.001031706
## 8  0.003850842 0.003542728 0.002061846 0.0005406360 0.000974644
```

```
## 9  0.076209253 0.060737582 0.042588559 0.0085233963 0.016010863
## 10 0.065383437 0.047097640 0.038019333 0.0066275448 0.013361318
## 11 0.058773118 0.039465252 0.035193069 0.0054579098 0.010910802
## 12 0.057061970 0.036425993 0.034599392 0.0051080314 0.010718174
## 13 0.055855681 0.034892560 0.034154563 0.0048510458 0.009911611
## 14 0.055289756 0.033291025 0.033823448 0.0046181495 0.009667824
## 15 0.054648272 0.032385730 0.033791627 0.0045720743 0.009578920
## 16 0.054246740 0.032012544 0.033605362 0.0044542445 0.009418154
## 17 0.086227733 0.074045718 0.045422661 0.0096471555 0.018175905
## 18 0.074455926 0.059240991 0.039950399 0.0075759170 0.014771885
## 19 0.067613827 0.051281796 0.036236863 0.0061238102 0.013166387
## 20 0.065198063 0.047768835 0.035214771 0.0056695460 0.012873067
## 21 0.064126962 0.046528439 0.034460565 0.0052977770 0.012673442
## 22 0.063512708 0.045623252 0.034107980 0.0051135017 0.012224358
## 23 0.062730852 0.044707188 0.033698007 0.0049793836 0.012015823
## 24 0.062454601 0.044293851 0.033693876 0.0048933303 0.011799427
##       se_otros sample   n
## 1  0.004336405    MAS  50
## 2  0.002844230    MAS 100
## 3  0.002167815    MAS 200
## 4  0.001698118    MAS 300
## 5  0.001463433    MAS 400
## 6  0.001203336    MAS 500
## 7  0.001141289    MAS 600
## 8  0.001051971    MAS 700
## 9  0.016374939    1ST  50
## 10 0.012767193    1ST 100
## 11 0.011077838    1ST 200
## 12 0.010359791    1ST 300
## 13 0.009994166    1ST 400
## 14 0.009541637    1ST 500
## 15 0.009287501    1ST 600
## 16 0.009180336    1ST 700
## 17 0.018912318    2ST  50
## 18 0.014937827    2ST 100
## 19 0.011807596    2ST 200
## 20 0.011275323    2ST 300
## 21 0.010372764    2ST 400
## 22 0.010043335    2ST 500
## 23 0.009739402    2ST 600
## 24 0.009539999    2ST 700
```

2. Grafica los datos de la tabla: realiza una gráfica de paneles (con `facet_wrap()`), cada partido en un panel, en el eje horizontal grafica el tamaño de muestra y en el eje vertical el error estándar, tendrás en una misma gráfica tres curvas, una para muestreo aleatorio simple y una para cada estratificación.

```
new_table <- tablaSe %>%
  gather(key =partido,value=se,se_pri:se_otros)

new_table %>%
  ggplot(aes(x=n,y=se))+
    geom_line(aes(x=n,y=se,colour=sample))+
    facet_wrap(~partido)+
    labs(title="Gráfica 2.1 Errores Estándar para cada tipo de muestreo",y="Error Estandar",x="Tamaño d
    theme(axis.text = element_text(angle=45,hjust=1,size=5)) +
```

```
  theme_light()
```



3. ¿Qué diseño y tamaño de muestra elegirías? Explica tu respuesta y de ser necesario repite los pasos i-iii para otros valores de n.

Si nos fueramos por elegir el diseño que minimiza el error estándar se escogería el diseño de muestreo aleatorio simple(MAS), dado que en la gráfica 2.1 es el método que tiene menor error en los 6 partidos. Esto también se pudo generar porque las muestras de cada estrato no son muy homogéneos, por lo que MAS es más preciso que las estratificaciones y si nos fueramos por la estratificación no sería una buena extrapolación de la población.

3. MCMC

Siguiendo con el conteo rápido de Guanajuato, calcularás intervalos de confianza usando el modelo propuesto en @mendoza2016.

Los autores proponen ajustar un modelo de manera independiente para cada candidato en cada estrato:

- Verosimilitud

$$X_{ij}^k|\theta_{ij},\tau_{ij} \sim N\left(n_i^k\theta_{ij}, \frac{\tau_{ij}}{n_i^k}\right)$$

para $k = 1, ..., c_i$, $i = 1, ..., N$, $j = 1, ..., J$

- Iniciales

$$p(\theta_{i,j},\tau_{ij}) \propto \tau_{ij}^{-1} I(\tau_{ij} > 0) I(0 < \theta_{i,j} < 1)$$

- Posterior

$$p(\theta_{ij},\tau_{ij}|X_{ij}) \sim N\left(\theta_{ij}\left|\frac{\sum_{k=1}^{c_i} x_{ij}^k}{\sum_{k=1}^{c_i} n_i^k}, \tau_{ij}\sum_{k=1}^{c_i} n_i^k\right.\right) I(0<\theta_{ij}<1) \times Ga\left(\tau_{ij}\left|\frac{c_i-1}{2}, \frac{1}{2}\left[\sum_{k=1}^{c_i}\frac{(x_{ij}^k)^2}{n_i^k} - \frac{\left(\sum_{k=1}^{c_i} x_{ij}^k\right)^2}{\sum_{k=1}^{c_i} n_i^k}\right]\right.\right)$$

Implementa el modelo y estima los resultados electorales de Guanajuato con la muestra:

Reporta estimaciones puntuales (media posterior) e intervalos del 95% de credibilidad para cada candidato.

Primereo definimos la distribución inicial.

```
prior <- function(tau){
  function(theta){
    (1/tau)*ifelse(tau>0,1,0)*ifelse(theta>0&theta<1,1,0)
  }
}
```

Definamos la función para la verosimilitud.

```
likelihood <- function(x_j,n,tau,c){
  function(theta){
    pnorm(theta,tau*nk)*ifelse(theta>0&theta<1,1,0)*rgamma(tau,(x_j^2/nk - x_j^2/nk)/2)
  }
}
```

Con los datos electorales definimos la prior inicial y verosimilitud

Definamos la función posterior

```
sims_posterior <- function(x_j, n, n_sims = 200){
    a_gamma <- (sum(n) - 1)/2
    print(a_gamma)
    b_gamma <- 1/ 2 * (sum(x_j ^ 2 / n) - sum(x_j) ^ 2 / sum(n))
    b_gamma <- ifelse(b_gamma == 0, 0.05, b_gamma)
    print(b_gamma)
    tau_sims <- rgamma(n_sims, shape = a_gamma, rate = b_gamma)
    print(tau_sims)
    media_normal <- sum(x_j) / sum(n)
    desv_normal <- map_dbl(tau_sims, ~ sqrt(1/(.*sum(n))))
    theta_sims <- map_dbl(desv_normal, ~rtnorm(1, media_normal, ., 0, 1))
    list(theta = theta_sims, tau = tau_sims)
}
```

La tabla de las muestras es la siguiente

```
gto_stratum_sizes <- gto_muestra %>%
    group_by(distrito_loc_17) %>%
    summarise(n_stratum = n()) %>%
    rename(strata = distrito_loc_17)

# gto_sample <- gto_muestra %>%
#   group_by(distrito_loc_17) %>%
#   sample_frac(0.06, replace = FALSE) %>%
#   ungroup()
```

Despivotemos la tabla para facilitar la manipulación

```
data_long <- gto_muestra %>%
        mutate(casilla_id = 1:n()) %>%
        rename(strata = distrito_loc_17) %>%
        select(casilla_id, strata, ln_total, pri_pvem:otros) %>%
        pivot_longer(pri_pvem:otros, names_to = 'party', values_to = 'votes')
```

```
# data_stratum <- gto_stratum_sizes %>%
#         rename(strata = distrito_loc_17)
```

Ahora podemos simular las tethas

```
theta_sims <- data_long %>%
  split(list(.$strata, .$party)) %>%
  map_df(~sims_posterior(x_j = .$votes, n = .$ln_total, n_sims = 100)$theta) %>%
  mutate(n_sim = 1:dplyr::n()) %>%
  gather(estrato_partido, theta, -n_sim) %>%
  separate(estrato_partido, into = c("strata", "party"),
           sep = "[.]") %>%
  mutate(strata = as.numeric(strata)) %>%
  left_join(gto_stratum_sizes, by = "strata")
```

Finalmente podemos simular las lambdas.

```
lambdas <- theta_sims %>%
        group_by(party, n_sim) %>%
        summarise(theta_wgt = sum(n_stratum * theta / sum(n_stratum))) %>%
        group_by(n_sim) %>%
        mutate(lambda = 100 * theta_wgt / sum(theta_wgt)) %>%
        ungroup() %>%
        select(-theta_wgt)
```

Las estimaciones puntuales e intervalos de 95% son:

```
lambdas %>%
        group_by(party) %>%
        summarise(
            mean_post = mean(lambda),
            median_post = median(lambda),
            std_error = sd(lambda),
            q_low = quantile(lambda, 0.025),
            q_sup = quantile(lambda, 0.975)
        ) %>%
        ungroup()
```

```
## # A tibble: 6 x 6
##   party    mean_post median_post std_error  q_low  q_sup
##   <chr>        <dbl>       <dbl>     <dbl>  <dbl>  <dbl>
## 1 mc           0.867       0.867   0.00438  0.859  0.876
## 2 otros        3.64        3.64    0.00444  3.63   3.65
## 3 pan_na      48.0        48.0     0.0185  47.9   48.0
## 4 prd          5.00        5.00    0.00721  4.99   5.01
## 5 pri_pvem    41.3        41.3     0.0184  41.3   41.4
## 6 pt           1.18        1.18    0.00221  1.17   1.18
```

Este ejercicio está basado complatamente en el desarrollado por Teresa Ortiz (tereom/quickcountmx).

4. Modelos jerárquicos, Stan y evaluación de ajuste

Implementación

1. **Modelo**. Ajusta el modelo y revisa convergencia, describe cuantas cadenas, iteraciones y etapa de calentamiento elegiste, además escribe como determinaste convergencia.

Primero definimos el modelo.

```
model_mrp <- stan_model('model_mrp.stan')
```

Posteriormente se corre el modelo con los datos.

```
model_mrp_fit <- sampling(model_mrp,
                          data = data_list,
                          chains = 3,
                          warmup = 500,
                          iter = 1500)
```

```
##
## SAMPLING FOR MODEL 'model_mrp' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 1: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 1: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 1: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 1: Iteration:  501 / 1500 [ 33%]  (Sampling)
## Chain 1: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 1: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 1: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 1: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 1: Iteration: 1250 / 1500 [ 83%]  (Sampling)
## Chain 1: Iteration: 1400 / 1500 [ 93%]  (Sampling)
## Chain 1: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 12.257 seconds (Warm-up)
## Chain 1:                20.022 seconds (Sampling)
## Chain 1:                32.279 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_mrp' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 2: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 2: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 2: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 2: Iteration:  501 / 1500 [ 33%]  (Sampling)
```

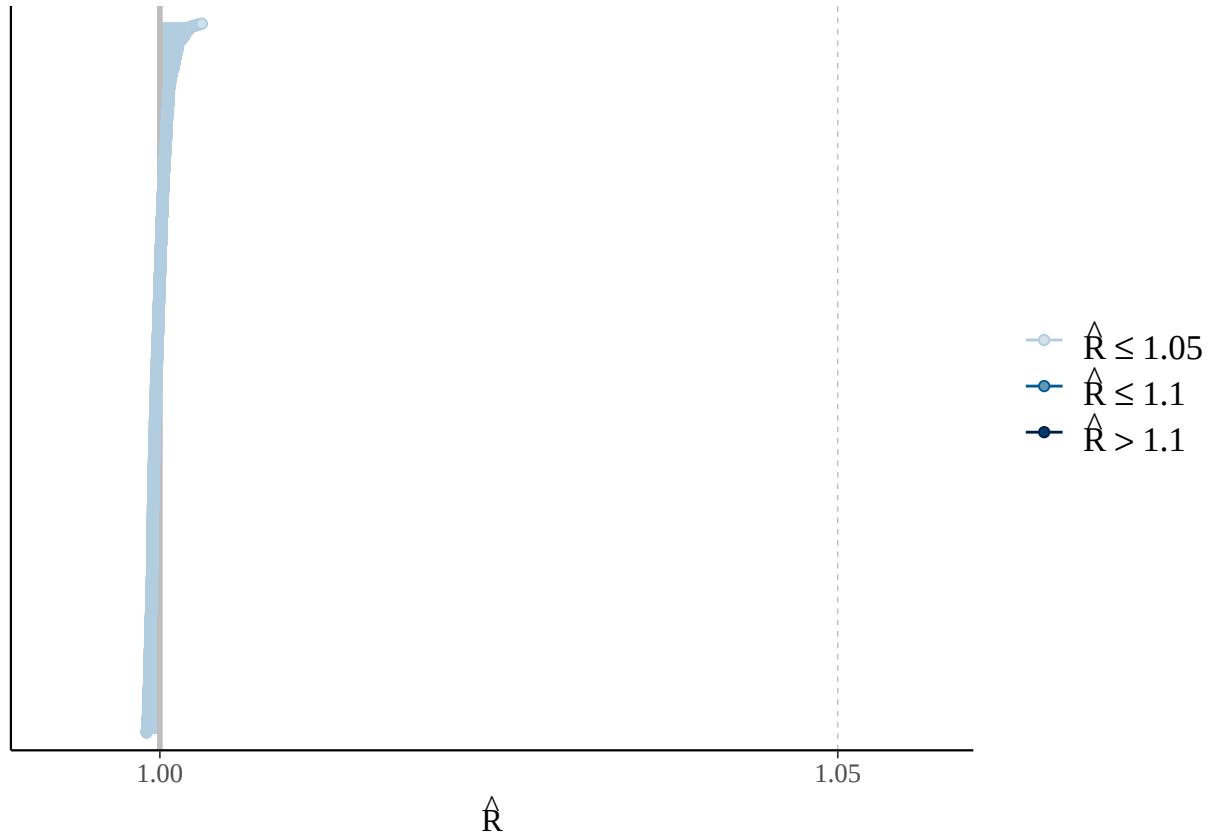
```
## Chain 2: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 2: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 2: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 2: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 2: Iteration: 1250 / 1500 [ 83%]  (Sampling)
## Chain 2: Iteration: 1400 / 1500 [ 93%]  (Sampling)
## Chain 2: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 12.631 seconds (Warm-up)
## Chain 2:                17.515 seconds (Sampling)
## Chain 2:                30.146 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_mrp' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 1500 [  0%]  (Warmup)
## Chain 3: Iteration:  150 / 1500 [ 10%]  (Warmup)
## Chain 3: Iteration:  300 / 1500 [ 20%]  (Warmup)
## Chain 3: Iteration:  450 / 1500 [ 30%]  (Warmup)
## Chain 3: Iteration:  501 / 1500 [ 33%]  (Sampling)
## Chain 3: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 3: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 3: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 3: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 3: Iteration: 1250 / 1500 [ 83%]  (Sampling)
## Chain 3: Iteration: 1400 / 1500 [ 93%]  (Sampling)
## Chain 3: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 13.924 seconds (Warm-up)
## Chain 3:                18.571 seconds (Sampling)
## Chain 3:                32.495 seconds (Total)
## Chain 3:
```

Primero evaluamos usando *traceplot*.

```
traceplot(model_mrp_fit)
```

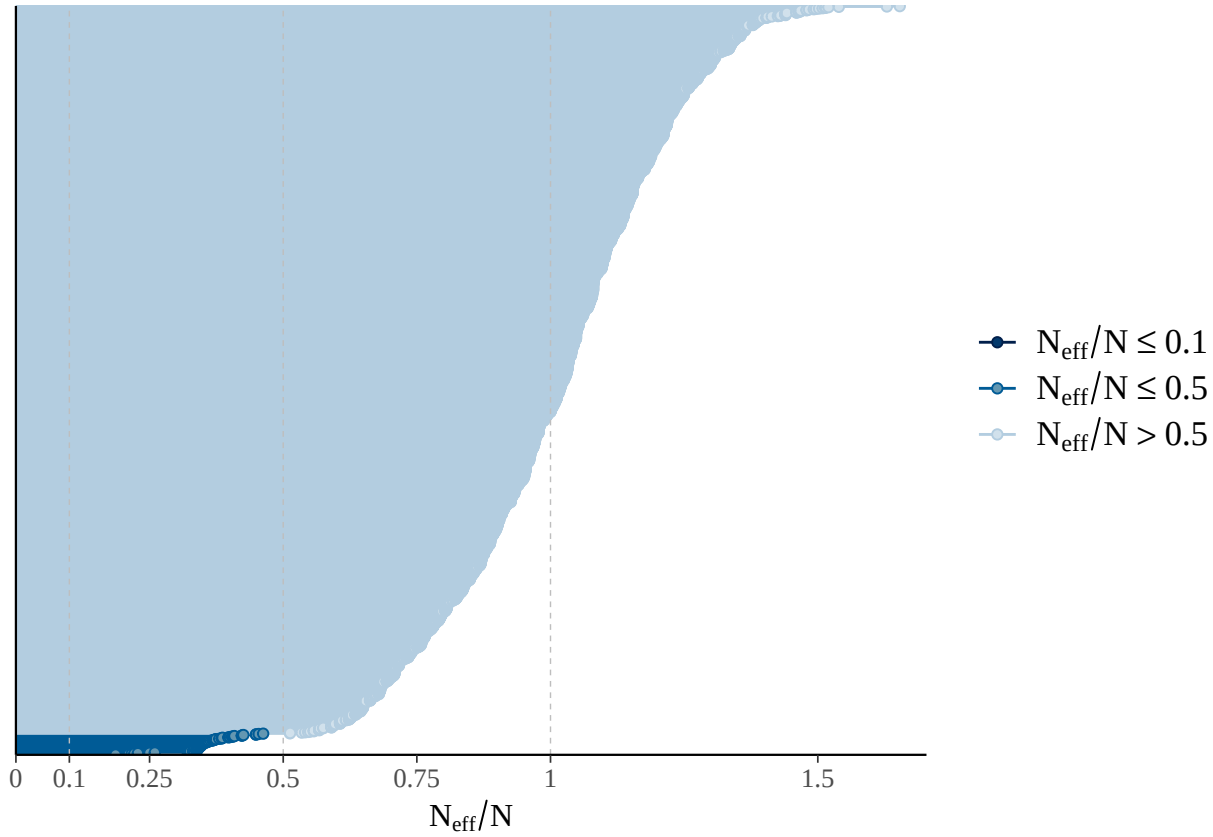
En todos los cassos los parmámetros parecen converger. Igualmente podemos revisar el diagnóstico de convergencia Gelman-Rubin.

```
mcmc_rhat(rhat(model_mrp_fit))
```



En todos los casos $\hat{R}$ es menor a 1.1 por lo que se puede comprobar la convergencia. Ahora revisemos el tamaño efectivo de la muestra.

```
mcmc_neff(neff_ratio(model_mrp_fit))
```



Este indicador es equivalente al visto en clase pero queremos que sea mayor a 0.1, por lo que se puede confiar en estimaciones estables. De esta manera se eligieron las corridas y calemntamientos mínimos para lograr estos resultados.

2. **Evaluación de ajuste**. Usaremos la distribución predictiva posterior para simular de modelo y comparar con los datos observados. En particular veremos como se comparan las simulaciones del modelo por estado, la gráfica con los datos será la que sigue:

Debes simular del modelo 10 conjutos de datos del tamaño de los originales (replicaciones de los datos) y hacer una gráfica de páneles donde muestres los datos originales y las replicaciones, ¿que concluyes al ver la gráfica?

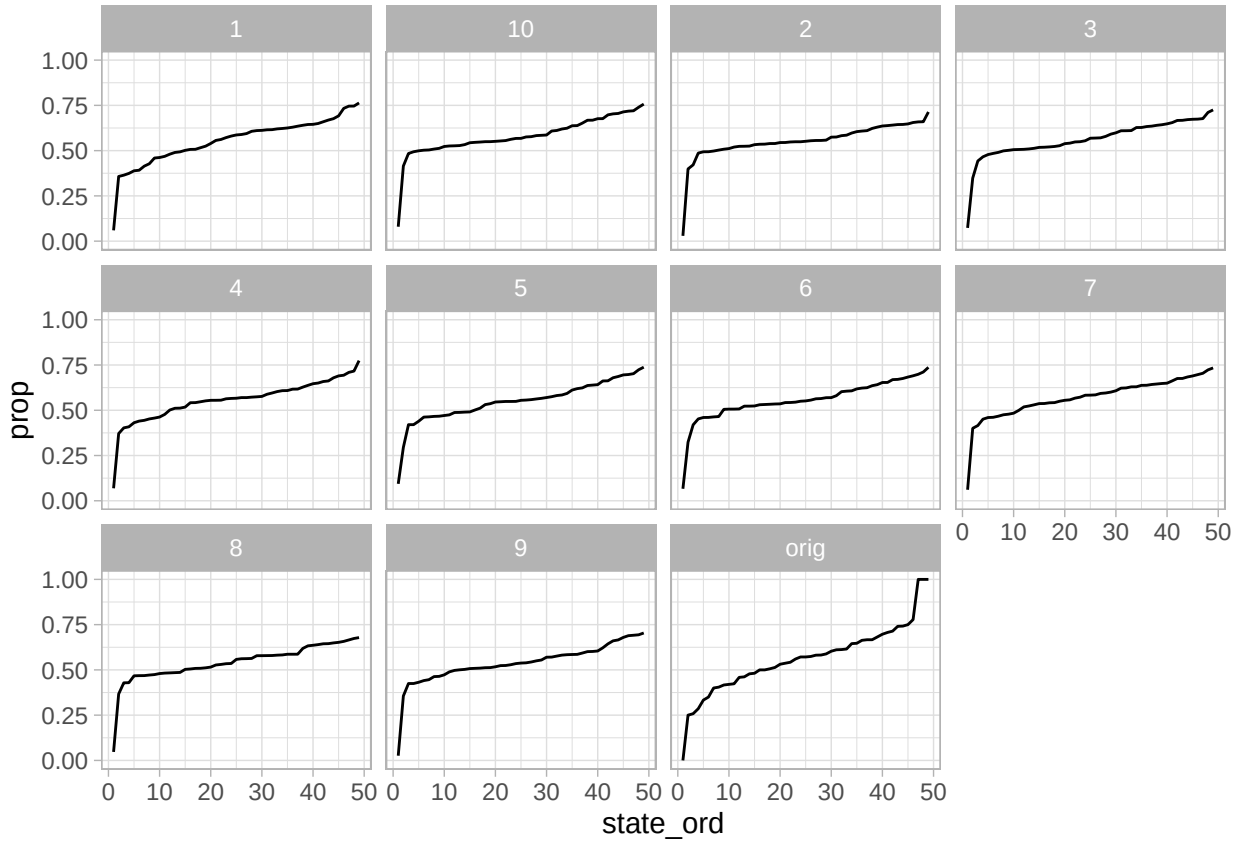
Primero obtenemos los parámetros *reg_pred* y los acomodamos en una tabla.

```
state <- tibble(id = 1:2015, state = last_poll$state)

par_sim <- as.data.frame(model_mrp_fit, pars = 'reg_pred') %>%
  mutate(n_sim = 1:n()) %>%
  filter(n_sim <= 10) %>%
  pivot_longer(cols = -n_sim, names_to = "id", values_to = "reg_pred", names_prefix = 'reg_pred') %>%
  left_join(state, by = 'id') %>%
  mutate(y = invlogit(reg_pred)) %>%
  group_by(n_sim, state) %>%
  summarise(prop = mean(y)) %>%
  ungroup() %>%
  arrange(n_sim, prop) %>%
  mutate(state_ord = rep(1:49,10))
```

```
bush_state$n_sim <- 'orig'

par_sim <- rbind(par_sim, bush_state)
```

Posteriormente hacemos la gráfica.

```
ggplot(par_sim, aes(x = state_ord, y = prop)) +
  geom_line() +
  facet_wrap(~n_sim) +
  theme_light()
```



3. El siguiente código predice para cada celda de la tabla del censo, vale la pena notar, que para cada celda tenemos una lista en el vector `pred` con las simuaciones que le corresponden.

Para hacer las estimaciones por estado hace falta ponderar por el número de casos en cada celda:

$$\theta_s = \frac{\sum_{j \in s} N_j \pi_j}{\sum_{j \in s} N_j}$$

4. Genera las simulaciones de θ_s, recuerda que debarás calcular una simulación de cada θ_s por cada simulación de π_j obtenida con el código de arriba. Realiza una gráfica con intervalos de credibilidad del 95% para cada $theta_s$.

Primero construimos una función que ayudará a obtener las tetas para cada simulación

```
f_theta <- function(i){
  pred_cell %>%
    mutate(Num = N*pred[i]) %>%
```

```
    group_by(state) %>%
    summarise(theta_s = sum(Num)/sum(N))
}
```

Ahora obtenemos las thetas

```
theta <- map(1:3000,f_theta) %>%
  bind_rows(.id='sim') %>%
  group_by(state) %>%
  summarise(
    media = mean(theta_s),
    int_l = quantile(theta_s,0.025),
    int_u = quantile(theta_s,0.975)
  )
```

Por último realizamosla gráfica de intervalos por estado.

```
ggplot(theta, aes(x = reorder(state,media), y = media, color = factor(state))) +
  geom_pointrange(aes(ymin=int_l, ymax = int_u), size = 0.3) +
  geom_point(color = 'black') +
  theme(legend.position = "none") +
  theme_light()
```