Examen Final

Bruno C. Gonzalez

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

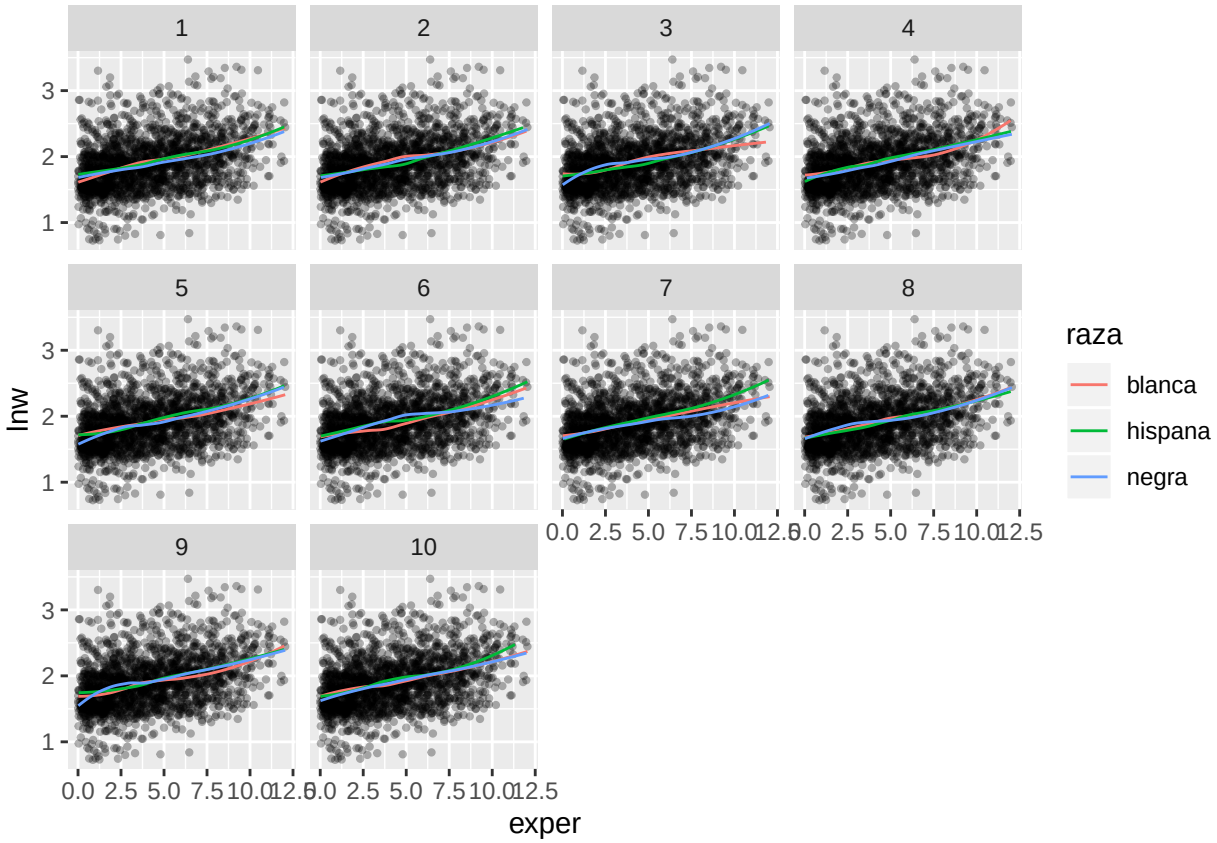
- El escenario nulo consiste en que no hay diferencia entre las razas. Para generar los datos nulos, la etiqueta de raza de cada sujeto se permuta, es decir, se reasigna la raza de cada sujeto de manera aleatoria (para todas las mediciones de un sujeto dado se reasigna una misma raza). Genera 9 conjuntos 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 = 0.3, 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. La persona no pudo distinguir los datos

- ¿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? Una persona por lo que el valor p es de 0.

2. Simulación para el cálculo de tamaños de muestra

3. MCMC

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 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 17.899 seconds (Warm-up)
## Chain 1:                39.441 seconds (Sampling)
## Chain 1:                57.34 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'model_mrp' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.001 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 10 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)
```

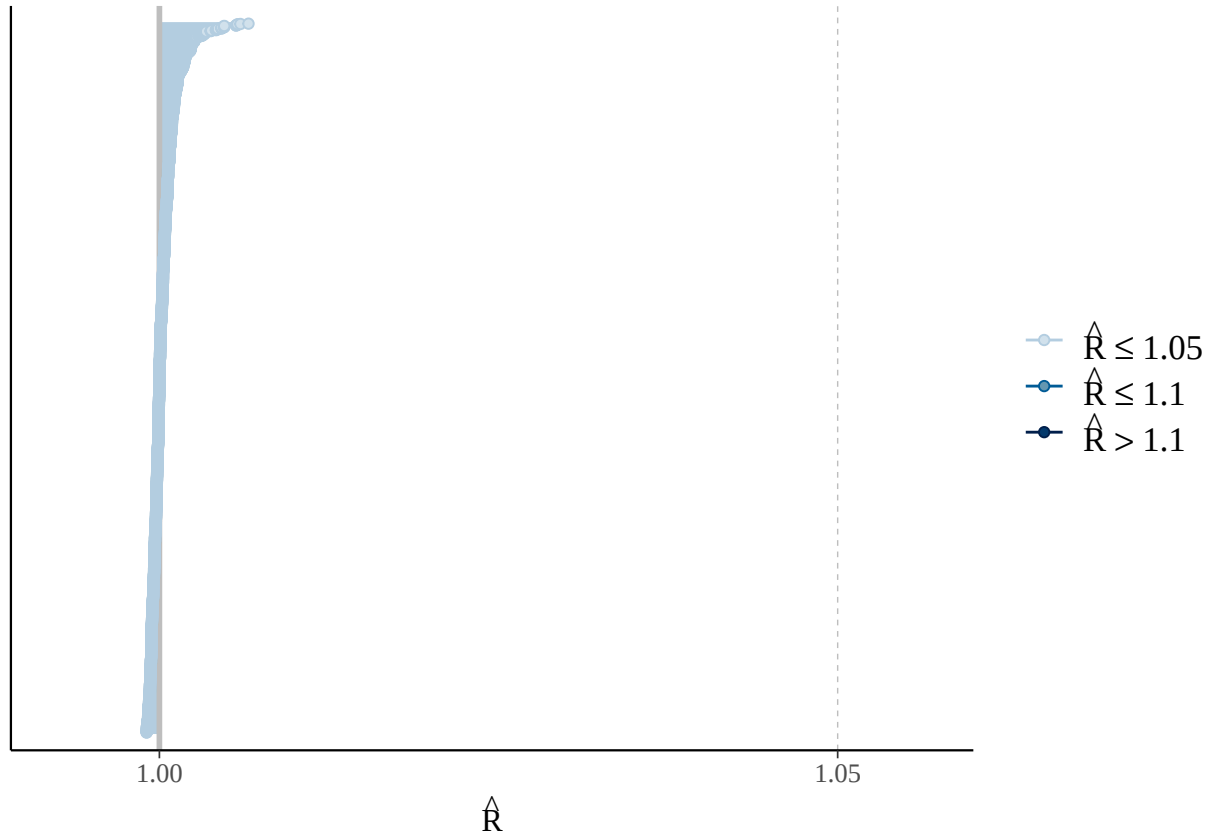
```
## Chain 2: Iteration:  650 / 1500 [ 43%]  (Sampling)
## Chain 2: Iteration:  800 / 1500 [ 53%]  (Sampling)
## Chain 2: Iteration:  950 / 1500 [ 63%]  (Sampling)
## Chain 2: Iteration: 1100 / 1500 [ 73%]  (Sampling)
## Chain 2: Iteration: 1250 / 1500 [ 83%]  (Sampling)
## Chain 2: Iteration: 1400 / 1500 [ 93%]  (Sampling)
## Chain 2: Iteration: 1500 / 1500 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 16.796 seconds (Warm-up)
## Chain 2:                30.21 seconds (Sampling)
## Chain 2:                47.006 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'model_mrp' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.001 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 10 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: 19.061 seconds (Warm-up)
## Chain 3:                34.486 seconds (Sampling)
## Chain 3:                53.547 seconds (Total)
## Chain 3:
```

Primero evaluamos usando *traceplot*.

```
traceplot(model_mrp_fit)
```

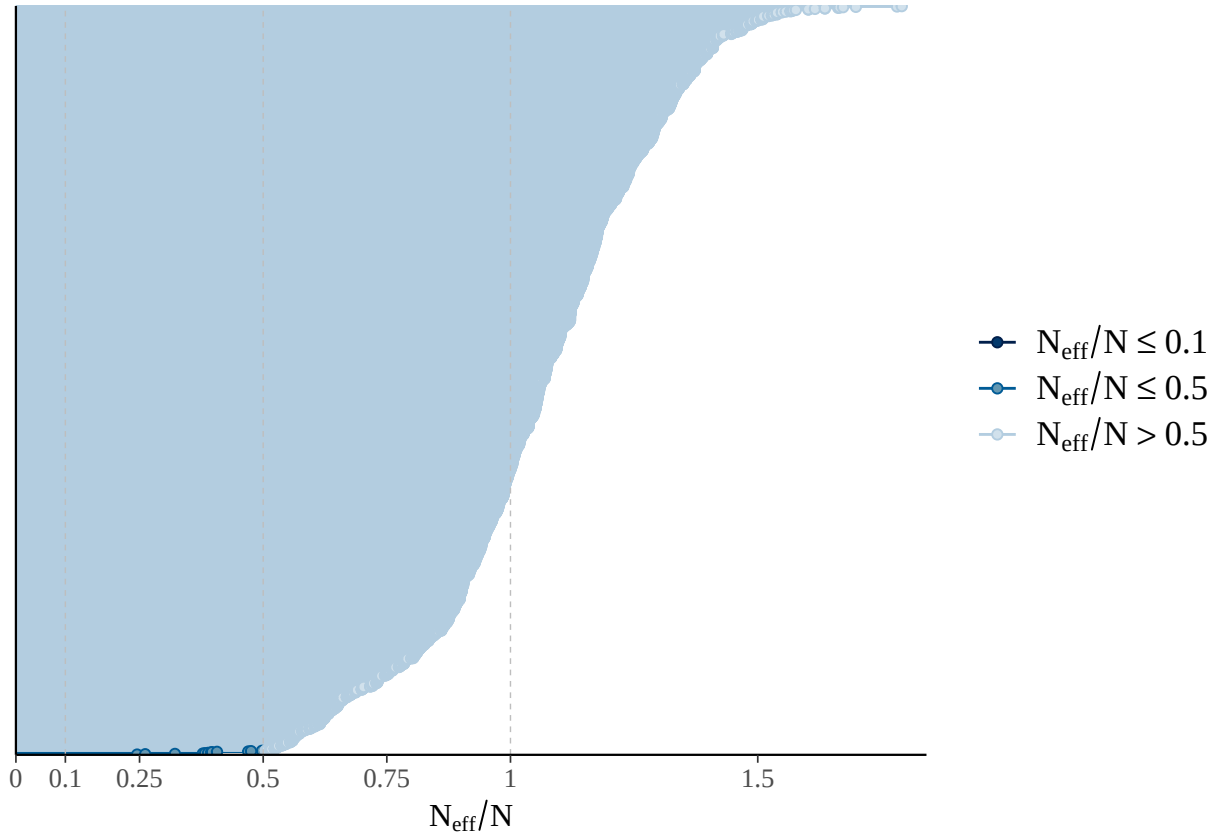
En todos los cassos los parmámetros parecen converger. Igualmente podemos revisar el diagnóstico de convergencia Gelman-Rubin.

```
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) %>%
```

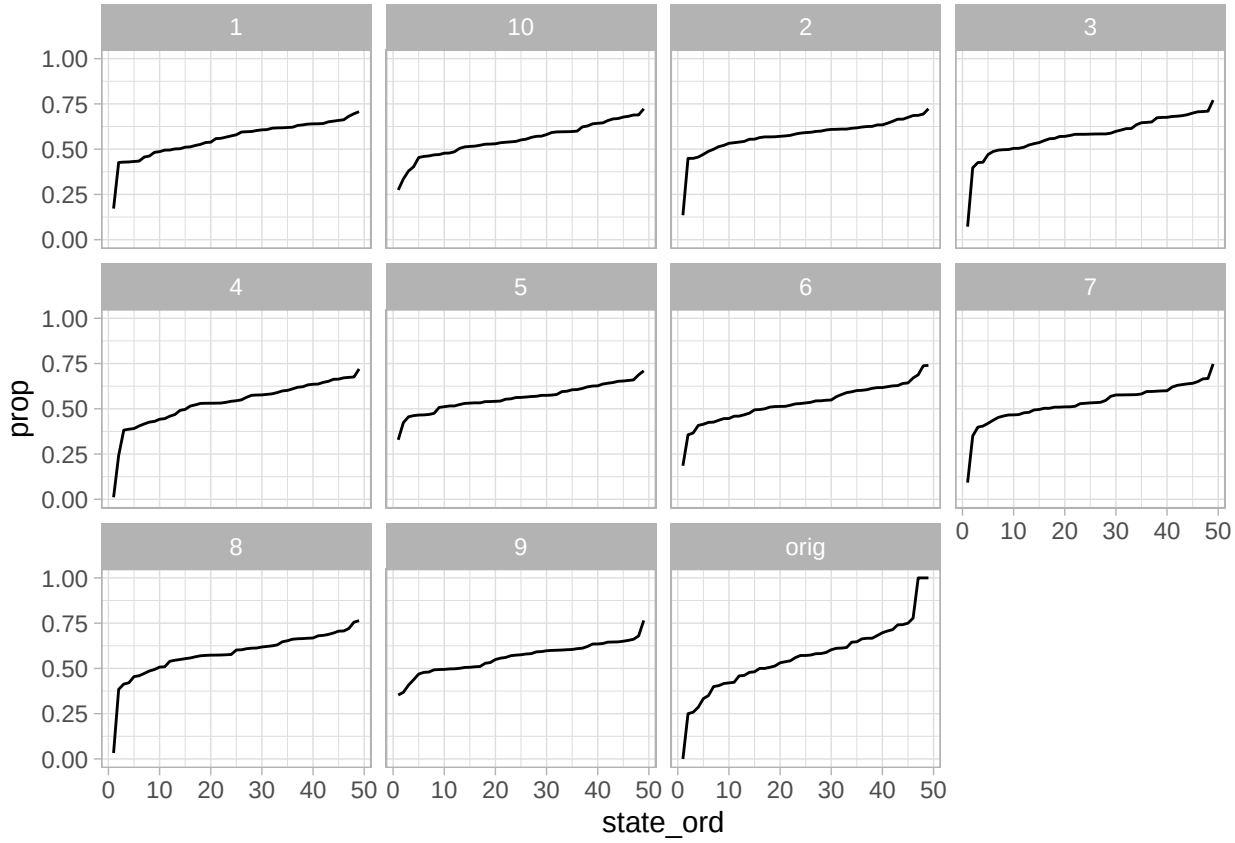
```
  mutate(state_ord = rep(1:49,10))

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

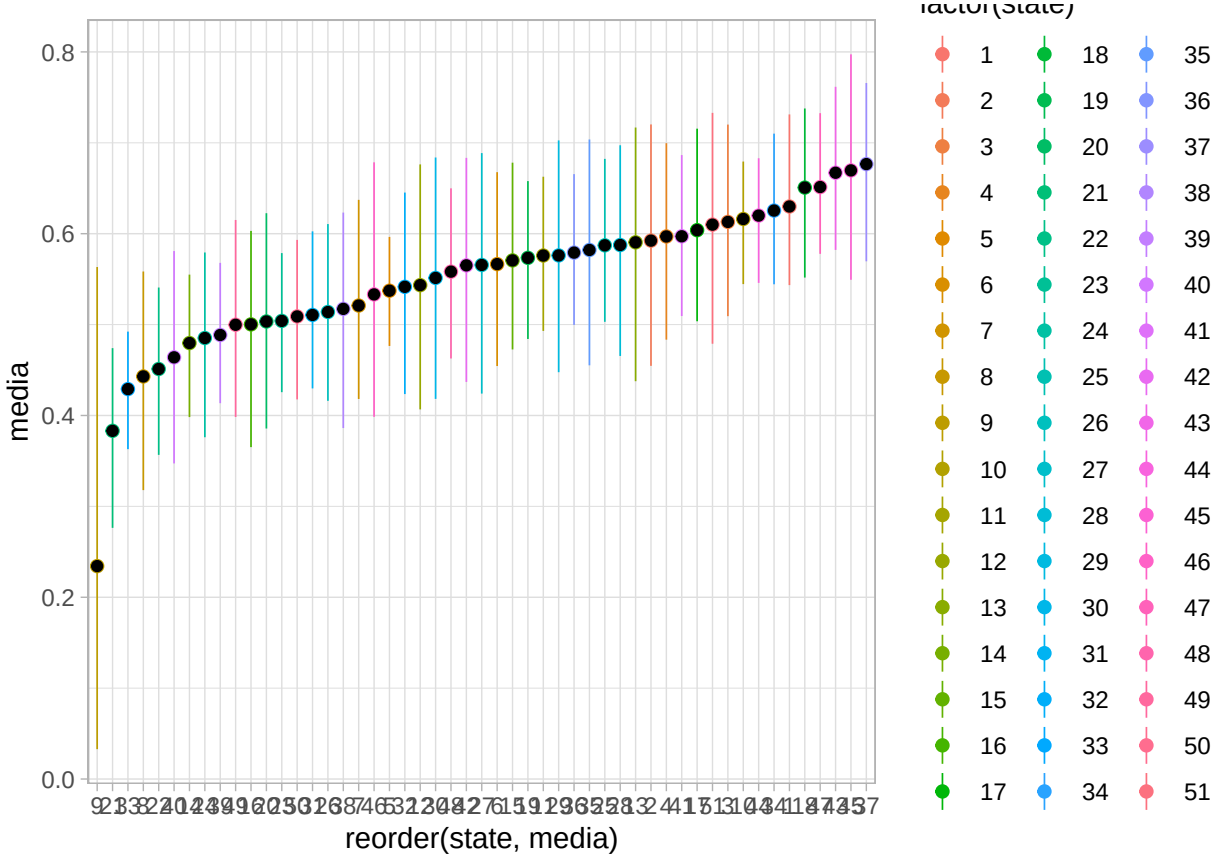
```
f_theta <- function(i){
  pred_cell %>%
    mutate(Num = N*pred[i]) %>%
    group_by(state) %>%
    summarise(theta_s = sum(Num)/sum(N))
}
```

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()
```


media
0.0
0.2
0.4
0.6
0.8
reorder(state, media)
factor(state)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51