

Inferencia e introducción a los modelos estadísticos con R

Curso intersemestral de Invierno - Posgrado de Ciencias Políticas y
Sociales

Ana Escoto

1/9/23

Table of contents

Introducción al curso	4
Objetivo general	4
Temas	4
Metodología	5
Proyecto en posit.cloud	5
Versiones anteriores	6
Facilitadora	6
Ana Ruth Escoto Castillo	6
Instalación de R y Rstudio	7
Introducción a R	7
Instalación en OS	7
Instalación en PC	7
Ojo	7
1 Introducción: {dplyr}	8
1.1 Paquetes	8
1.2 Cargando los datos	8
1.3 Un poquito de {dplyr} y limpieza	9
1.3.1 Primero, los pipes	9
1.3.2 Limpieza de nombres con {janitor}	10
1.4 <code>select()</code> y <code>filter()</code>	16
1.5 Tabulados con <code>tabyl()</code>	17

1.5.1	Cálculo de frecuencias	19
1.5.2	Totales y porcentajes	19
2	Repaso de visualización de datos y {ggplot2}	23
2.1	Paquetes	23
2.2	Cargando los datos	24
2.3	<i>Grammar of tables: gt</i>	24
2.4	Descriptivos para variables cuantitativas	26
2.4.1	Medidas numéricas básicas	26
2.5	Visualización de datos, un pequeño disclaimer	27
2.5.1	Gráficos de base	27
2.6	<i>Grammar of graphics: ggplot</i>	29
2.6.1	Un lienzo para dibujar	29
2.6.2	Gráficos univariados	30
2.7	Intro a dos variables	34
3	Introducción a la inferencia	37
3.1	Paquetes	37
3.2	Cargando los datos	38
3.3	Hipótesis e intervalos de confianza	38
3.3.1	t-test	38
3.3.2	Enchulando un poquito	41
3.3.3	Prueba para proporción	43
3.4	Análisis de varianza	45
3.4.1	Primero un gráfico	45
3.4.2	Comparación entre grupos	47
3.4.3	Supuestos de ANOVA	47
3.4.4	Kruskal-Wallis test	48
3.5	Factores de expansión	51
3.5.1	La función tally	51
3.6	Otras formas	52

Material anexo	57
Lista general de YouTube	57
Scripts	57
Cuestionarios	57
Primer cuestionario	57
Sesión 1	57
Video	57
Cheat Sheets	57

Introducción al curso

Objetivo general

Que el estudiantado sea capaz de realizar inferencia estadística y modelado de una variable dependiente utilizando R aplicado a las bases de datos mexicanas.

Temas

1. Revisión de elementos estadísticos básicos desde “tidyverse”

- a. Tablas de múltiples entradas
- b. Repaso de ggplot2

2. Pruebas de hipótesis e intervalos de confianza

- a. De una sola media
- b. De dos medias
- c. Medias apareadas
- d. Proporciones
- e. Diferencia de proporciones
- f. Chi-cuadrado de independencia
- g. Prueba ANOVA de un solo factor
- h. Pruebas no paramétricas

3. Factores de expansión y diseño muestral complejo

- a. De “survey” a “srvyr”
- b. Tabulados
- c. Intervalos de confianza para medias y cuantiles

4. Introducción al modelo de regresión lineal

- a. Simple
- b. Múltiple
- c. Evaluación de supuestos

5. Introducción a los modelos lineales generalizados

- a. Introducción a la regresión logística
- b. Evaluación de supuestos
- c. Efectos marginales
- d. Interacciones y efectos de más de primer orden

Metodología

La metodología del curso consistirá en lo siguiente:

1. *La exposición de la facilitadora.* Durante la primera parte de la sesión, se expondrán los comandos necesarios para llevar a cabo cada tema. Se dará una introducción sobre la temática y se buscará dar ejemplos concretos para facilitar el aprendizaje. Se espera que el personal exponga sus dudas o comentarios a lo largo de la explicación.
2. *Realización de ejercicios prácticos.* Al final de cada sesión, corresponderá a las personas asistentes del curso realizar individualmente o en parejas un ejercicio relacionado con lo visto en la primera parte de la clase.
3. *Consulta autónoma de material.* Tanto la exposición como los ejercicios serán acompañado de material de consulta realizado ad hoc para el curso y el contenido, de tal manera que el estudiantado pueda volver a los códigos y las explicaciones posteriormente.

Proyecto en posit.cloud

<https://posit.cloud/content/5191591>

Con este proyecto se trabajará a lo largo del curso

Versiones anteriores

Esta es la tercera vez que se imparte el curso. Las versiones anteriores están

- https://aniuxa.github.io/posgrado_modelo/
- https://aniuxa.github.io/posgrado_modelo2/

Facilitadora

Ana Ruth Escoto Castillo

Doctora en Estudios de Población. Centro de Estudios Demográficos y Urbanos, El Colegio de México.

Semblanza

Profesora de tiempo completo en la Facultad de Ciencias Políticas y Sociales. Investigadora nivel I en el Sistema Nacional de Investigadores. Maestra en Población y Desarrollo por la Facultad Latinoamericana de Ciencias Sociales (FLACSO) – Sede México. Posee experiencia en recolección de información estadística, diseño y control de procesos de recolección y su procesamiento. Ha aplicado diversos métodos y herramientas multivariadas, homologación de información y comparabilidad de fuentes en sus investigaciones, así como usa de diversos softwares estadísticos, y ha impartido clases de estadística aplicada a nivel de licenciatura y posgrado. Es co-coordinadora del Capítulo de CDMX de la iniciativa RLadies.

Instalación de R y Rstudio

Introducción a R

<https://youtu.be/YkN5urybh2A>

Instalación en OS

<https://youtu.be/icWV8jzYOtA>

Instalación en PC

<https://youtu.be/TNSQikMfgJI>

Ojo

Desde octubre de 2022, RStudio se volvió “**Posit**”

Chapter 1

Introducción: {dplyr}

1.1 Paquetes

```
if (!require("pacman")) install.packages("pacman")#instala pacman si se requiere
```

Loading required package: pacman

```
pacman::p_load(tidyverse,  
               readxl,  
               writexl,  
               haven,  
               sjlabelled,  
               janitor,  
               infer,  
               ggpubr,  
               magrittr,  
               gt)
```

1.2 Cargando los datos

Desde STATA

```
tlaxt322<- haven::read_dta("./datos/tlaxt322.dta")
```

Desde Excel:

```
ICI_2021 <- readxl::read_excel("./datos/ICI_2021.xlsx",
                               sheet = "para_importar")
```

New names:

```
* `` -> `...2`
```

1.3 Un poquito de {dplyr} y limpieza

1.3.1 Primero, los pipes

R utiliza dos pipes el nativo `|>` y el pipe que está en `{dplyr}` `%>%`. Algunas de las diferencias las puedes checar acá <https://eliocamp.github.io/codigo-r/2021/05/r-pipa-nativa/>

En estas prácticas utilizaremos el segundo, pero son muy parecidos y para que esta instructora recicle algunos de sus códigos viejos. Pero funcionan igual:

```
tlaxt322|> #pipe nativo, no necesita instalación
  head()
```

```
# A tibble: 6 x 114
  r_def      loc  mun  est est_d~1 est_d~2 ageb t_loc~3 t_loc~4 cd_a
  <dbl+lbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
1 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
2 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
3 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
4 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
5 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
6 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
# ... with 104 more variables: ent <dbl+lbl>, con <dbl>, upm <dbl>,
#   d_sem <dbl+lbl>, n_pro_viv <dbl>, v_sel <dbl+lbl>, n_hog <dbl+lbl>,
#   h_mud <dbl+lbl>, n_ent <dbl+lbl>, per <dbl+lbl>, n_ren <dbl+lbl>,
#   c_res <dbl+lbl>, par_c <dbl>, sex <dbl+lbl>, eda <dbl>, nac_dia <dbl+lbl>,
#   nac_mes <dbl+lbl>, nac_anio <dbl>, l_nac_c <dbl+lbl>, cs_p12 <dbl+lbl>,
#   cs_p13_1 <dbl+lbl>, cs_p13_2 <dbl+lbl>, cs_p14_c <chr>, cs_p15 <dbl+lbl>,
#   cs_p16 <dbl+lbl>, cs_p17 <dbl+lbl>, n_hij <dbl+lbl>, e_con <dbl+lbl>, ...
```

```
tlaxt322 %>% #pipe de dplyr, necesita instalación de dplyr en tidyverse
  head()
```

```
# A tibble: 6 x 114
  r_def      loc  mun  est est_d~1 est_d~2 ageb t_loc~3 t_loc~4 cd_a
<dbl+lbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl+1> <dbl+1> <dbl+1b>
1 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
2 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
3 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
4 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
5 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
6 0 [Entrevist~ NA    31    20    413      NA    0 2 [Loc~ NA    39 [Tla~
# ... with 104 more variables: ent <dbl+lbl>, con <dbl>, upm <dbl>,
#   d_sem <dbl+lbl>, n_pro_viv <dbl>, v_sel <dbl+lbl>, n_hog <dbl+lbl>,
#   h_mud <dbl+lbl>, n_ent <dbl+lbl>, per <dbl+lbl>, n_ren <dbl+lbl>,
#   c_res <dbl+lbl>, par_c <dbl>, sex <dbl+lbl>, eda <dbl>, nac_dia <dbl+lbl>,
#   nac_mes <dbl+lbl>, nac_anio <dbl>, l_nac_c <dbl+lbl>, cs_p12 <dbl+lbl>,
#   cs_p13_1 <dbl+lbl>, cs_p13_2 <dbl+lbl>, cs_p14_c <chr>, cs_p15 <dbl+lbl>,
#   cs_p16 <dbl+lbl>, cs_p17 <dbl+lbl>, n_hij <dbl+lbl>, e_con <dbl+lbl>, ...
```

1.3.2 Limpieza de nombres con {janitor}

Este paso también nos permitirá enseñar otro *pipe* que está en el paquete {magrittr}.

Los nombres de una base de datos son los nombres de las columnas.

```
names(tlaxt322)
```

```
[1] "r_def"      "loc"        "mun"        "est"        "est_d_tri"
[6] "est_d_men"  "ageb"       "t_loc_tri"  "t_loc_men"  "cd_a"
[11] "ent"        "con"        "upm"        "d_sem"      "n_pro_viv"
[16] "v_sel"      "n_hog"      "h_mud"      "n_ent"      "per"
[21] "n_ren"      "c_res"      "par_c"      "sex"        "eda"
[26] "nac_dia"    "nac_mes"    "nac_anio"   "l_nac_c"    "cs_p12"
[31] "cs_p13_1"   "cs_p13_2"   "cs_p14_c"   "cs_p15"     "cs_p16"
[36] "cs_p17"     "n_hij"      "e_con"      "cs_p20a_1"  "cs_p20a_c"
[41] "cs_p20b_1"  "cs_p20b_c"  "cs_p20c_1"  "cs_ad_mot"  "cs_p21_des"
[46] "cs_ad_des"  "cs_nr_mot"  "cs_p23_des" "cs_nr_ori"  "ur"
[51] "zona"       "salario"    "fac_tri"    "fac_men"    "clase1"
[56] "clase2"     "clase3"     "pos_ocu"    "seg_soc"    "rama"
[61] "c_ocu11c"   "ing7c"      "dur9c"      "emple7c"    "medica5c"
[66] "buscar5c"   "rama_est1"  "rama_est2"  "dur_est"    "ambito1"
[71] "ambito2"    "tue1"       "tue2"       "tue3"       "busqueda"
[76] "d_ant_lab"  "d_cexp_est" "dur_des"    "sub_o"      "s_clasifi"
[81] "remune2c"   "pre_asa"    "tip_con"    "dispo"      "nodispo"
[86] "c_inac5c"   "pnea_est"   "niv_ins"    "eda5c"      "eda7c"
[91] "eda12c"     "eda19c"     "hij5c"      "domestico"  "anios_esc"
```

```

[96] "hrsocup"      "ingocup"      "ing_x_hrs"    "tpg_p8a"      "tcco"
[101] "cp_anoc"       "imssissste"   "ma48me1sm"    "p14apoyos"    "scian"
[106] "t_tra"         "emp_ppal"     "tue_ppal"     "trans_ppal"   "mh_fil2"
[111] "mh_col"        "sec_ins"      "tipo"         "mes_cal"

```

```
names(ICI_2021)
```

```

[1] "País"
[2] "...2"
[3] "Protección de derechos humanos"
[4] "Homicidios dolosos"
[5] "Confianza en la policía"
[6] "Independencia del poder judicial"
[7] "Protección de derechos de propiedad"
[8] "Tiempo para resolver quiebras"
[9] "Cumplimiento de contratos"
[10] "Índice de Estado de Derecho"
[11] "Índice de Paz Global"
[12] "Contaminación del aire"
[13] "Emisiones de CO2"
[14] "Recursos hídricos renovables"
[15] "Áreas naturales protegidas"
[16] "Superficie forestal perdida"
[17] "Uso de pesticidas"
[18] "Fuentes de energía no contaminantes"
[19] "Índice de vulnerabilidad a efectos del cambio climático"
[20] "Índice de Gini"
[21] "Índice Global de Brecha de Género"
[22] "Mujeres en la PEA"
[23] "Dependientes de la PEA"
[24] "Acceso a agua potable"
[25] "Acceso a alcantarillado"
[26] "Analfabetismo"
[27] "Escolaridad promedio"
[28] "Calidad educativa"
[29] "Esperanza de vida"
[30] "Mortalidad infantil"
[31] "Cobertura de vacunación"
[32] "Médicos y médicas"
[33] "Gasto en salud per cápita"
[34] "Gasto en salud por cuenta propia"
[35] "Estabilidad política y ausencia de violencia"
[36] "Interferencia militar en el Estado de derecho o en el proceso político"
[37] "Libertades civiles"

```

- [38] "Índice de Percepción de Corrupción"
- [39] "Disponibilidad de información pública"
- [40] "Participación electoral"
- [41] "Equidad en los congresos"
- [42] "Índice de efectividad del gobierno"
- [43] "Miembro de la Alianza para el Gobierno Abierto"
- [44] "Índice de desarrollo de Gobierno Electrónico"
- [45] "Facilidad para abrir una empresa"
- [46] "Tiempo para preparar y pagar impuestos"
- [47] "Ingresos fiscales"
- [48] "Finanzas sanas"
- [49] "Carga impositiva"
- [50] "Edad efectiva de retiro"
- [51] "Flexibilidad de las leyes laborales"
- [52] "Productividad media del trabajo"
- [53] "Valor agregado de la industria"
- [54] "Índice de transparencia y regulación de la propiedad privada"
- [55] "Crecimiento del PIB"
- [56] "Crecimiento promedio del PIB"
- [57] "Inflación"
- [58] "Inflación promedio"
- [59] "Desempleo"
- [60] "Deuda externa"
- [61] "Calificación de deuda"
- [62] "Reservas"
- [63] "Libertad económica"
- [64] "Índice Riesgos de seguridad energética"
- [65] "Líneas móviles"
- [66] "Usuarios de internet"
- [67] "Servidores de internet seguros"
- [68] "Flujo de pasajeros aéreos"
- [69] "Índice de desempeño logístico (transporte)"
- [70] "Tráfico portuario de contenedores"
- [71] "Penetración del sistema financiero privado"
- [72] "Capitalización del mercado de valores"
- [73] "Socios comerciales efectivos"
- [74] "Apertura comercial"
- [75] "Diversificación de las exportaciones"
- [76] "Diversificación de las importaciones"
- [77] "Libertad comercial"
- [78] "Inversión extranjera directa (neta)"
- [79] "Inversión Extranjera Directa neta promedio"
- [80] "Ingresos por turismo"
- [81] "Gasto en investigación y desarrollo"
- [82] "Coeficiente de invención"
- [83] "Artículos científicos y técnicos"

```
[84] "Exportaciones de alta tecnología"
[85] "Índice de Complejidad Económica"
[86] "Empresas ISO 9001"
[87] "PIB en servicios"
[88] "0"
[89] "Inversión (FBCF)"
[90] "Talento"
```

Como vemos en las bases hay mayúsculas, caracteres especiales y demás. Esto lo podemos cambiar

```
ICI_2021<-ICI_2021 %>%
  janitor::clean_names()

names(ICI_2021)
```

```
[1] "pais"
[2] "x2"
[3] "proteccion_de_derechos_humanos"
[4] "homicidios_dolosos"
[5] "confianza_en_la_policia"
[6] "independencia_del_poder_judicial"
[7] "proteccion_de_derechos_de_propiedad"
[8] "tiempo_para_resolver_quiebras"
[9] "cumplimiento_de_contratos"
[10] "indice_de_estado_de_derecho"
[11] "indice_de_paz_global"
[12] "contaminacion_del_aire"
[13] "emisiones_de_co2"
[14] "recursos_hidricos_renovables"
[15] "areas_naturales_protegidas"
[16] "superficie_forestal_perdida"
[17] "uso_de_pesticidas"
[18] "fuentes_de_energia_no_contaminantes"
[19] "indice_de_vulnerabilidad_a_efectos_del_cambio_climatico"
[20] "indice_de_gini"
[21] "indice_global_de_brecha_de_genero"
[22] "mujeres_en_la_pea"
[23] "dependientes_de_la_pea"
[24] "acceso_a_agua_potable"
[25] "acceso_a_alcantarillado"
[26] "analfabetismo"
[27] "escolaridad_promedio"
[28] "calidad_educativa"
[29] "esperanza_de_vida"
```

[30] "mortalidad_infantil"
[31] "cobertura_de_vacunacion"
[32] "medicos_y_medicas"
[33] "gasto_en_salud_per_capita"
[34] "gasto_en_salud_por_cuenta_propia"
[35] "estabilidad_politica_y_ausencia_de_violencia"
[36] "interferencia_militar_en_el_estado_de_derecho_o_en_el_proceso_politico"
[37] "libertades_civiles"
[38] "indice_de_percepcion_de_corrupcion"
[39] "disponibilidad_de_informacion_publica"
[40] "participacion_electoral"
[41] "equidad_en_los_congresos"
[42] "indice_de_efectividad_del_gobierno"
[43] "miembro_de_la_alianza_para_el_gobierno_abierto"
[44] "indice_de_desarrollo_de_gobierno_electronico"
[45] "facilidad_para_abrir_una_empresa"
[46] "tiempo_para_preparar_y_pagar_impuestos"
[47] "ingresos_fiscales"
[48] "finanzas_sanas"
[49] "carga_impositiva"
[50] "edad_efectiva_de_retiro"
[51] "flexibilidad_de_las_leyes_laborales"
[52] "productividad_media_del_trabajo"
[53] "valor_agregado_de_la_industria"
[54] "indice_de_transparencia_y_regulacion_de_la_propiedad_privada"
[55] "crecimiento_del_pib"
[56] "crecimiento_promedio_del_pib"
[57] "inflacion"
[58] "inflacion_promedio"
[59] "desempleo"
[60] "deuda_externa"
[61] "calificacion_de_deuda"
[62] "reservas"
[63] "libertad_economica"
[64] "indice_riesgos_de_seguridad_energetica"
[65] "lineas_moviles"
[66] "usuarios_de_internet"
[67] "servidores_de_internet_seguros"
[68] "flujo_de_pasajeros_aereos"
[69] "indice_de_desempeno_logistico_transporte"
[70] "trafico_portuario_de_contenedores"
[71] "penetracion_del_sistema_financiero_privado"
[72] "capitalizacion_del_mercado_de_valores"
[73] "socios_comerciales_efectivos"
[74] "apertura_comercial"
[75] "diversificacion_de_las_exportaciones"

```

[76] "diversificacion_de_las_importaciones"
[77] "libertad_comercial"
[78] "inversion_extranjera_directa_neta"
[79] "inversion_extranjera_directa_neta_promedio"
[80] "ingresos_por_turismo"
[81] "gasto_en_investigacion_y_desarrollo"
[82] "coeficiente_de_invencion"
[83] "articulos_cientificos_y_tecnicos"
[84] "exportaciones_de_alta_tecnologia"
[85] "indice_de_complejidad_economica"
[86] "empresas_iso_9001"
[87] "pib_en_servicios"
[88] "x0"
[89] "inversion_fbcf"
[90] "talento"

```

Si quisiéramos que la acción quedará de un solo, podemos usar un pipe diferente:

```

tlaxt322%<>%
  clean_names()

names(tlaxt322)

```

```

[1] "r_def"      "loc"      "mun"      "est"      "est_d_tri"
[6] "est_d_men" "ageb"     "t_loc_tri" "t_loc_men" "cd_a"
[11] "ent"       "con"      "upm"      "d_sem"     "n_pro_viv"
[16] "v_sel"     "n_hog"    "h_mud"    "n_ent"     "per"
[21] "n_ren"     "c_res"    "par_c"    "sex"       "eda"
[26] "nac_dia"   "nac_mes"  "nac_anio" "l_nac_c"   "cs_p12"
[31] "cs_p13_1"  "cs_p13_2" "cs_p14_c" "cs_p15"    "cs_p16"
[36] "cs_p17"    "n_hij"    "e_con"    "cs_p20a_1" "cs_p20a_c"
[41] "cs_p20b_1" "cs_p20b_c" "cs_p20c_1" "cs_ad_mot" "cs_p21_des"
[46] "cs_ad_des" "cs_nr_mot" "cs_p23_des" "cs_nr_ori" "ur"
[51] "zona"      "salario"  "fac_tri"  "fac_men"  "clase1"
[56] "clase2"    "clase3"   "pos_ocu"  "seg_soc"  "rama"
[61] "c_ocu11c"  "ing7c"    "dur9c"    "emple7c"  "medica5c"
[66] "buscar5c"  "rama_est1" "rama_est2" "dur_est"  "ambito1"
[71] "ambito2"   "tue1"     "tue2"     "tue3"     "busqueda"
[76] "d_ant_lab" "d_cexp_est" "dur_des"  "sub_o"    "s_clasifi"
[81] "remune2c"  "pre_asa"   "tip_con"  "dispo"    "nodispo"
[86] "c_inac5c"  "pnea_est"  "niv_ins"  "eda5c"    "eda7c"
[91] "eda12c"    "eda19c"    "hij5c"    "domestico" "anios_esc"
[96] "hrsocup"   "ingocup"   "ing_x_hrs" "tpg_p8a"  "tcco"
[101] "cp_anoc"   "imssissste" "ma48me1sm" "p14apoyos" "scian"
[106] "t_tra"     "emp_ppal"  "tue_ppal" "trans_ppal" "mh_fil2"

```



```
[111] "mh_col"      "sec_ins"      "tipo"         "mes_cal"
```

Más de otros *pipes* <https://r4ds.had.co.nz/pipes.html>

1.4 `select()` y `filter()`

Este es un recordatorio de que en `{dplyr}`, se filtran CASOS, es decir, líneas o renglones, y se seleccionan VARIABLES.

Por ejemplo:

```
tlaxt322%>%
  dplyr::select(sex, eda) %>%
  dplyr::filter(eda>11)
```

```
# A tibble: 9,205 x 2
   sex      eda
  <dbl+lbl> <dbl>
1 1 [Hombre]   34
2 2 [Mujer]    57
3 1 [Hombre]   71
4 1 [Hombre]   67
5 2 [Mujer]    60
6 2 [Mujer]    35
7 1 [Hombre]   39
8 1 [Hombre]   38
9 2 [Mujer]    33
10 1 [Hombre]  14
# ... with 9,195 more rows
```

En la documentación de la base de datos de la ENOE se nos señala que debemos quedarnos con quienes tienen entrevista completa `r_def==0` y con quienes son habitante habituales (`c_res!=2`)

Hagamos estos cambios:

```
tlaxt322%<>%
  filter(r_def==0) %>%
  filter(!c_res==2)
```

1.5 Tabulados con `tabyl()`

El comando `tabyl` del paquete `{janitor}` nos sirve para hacer tabulados. Para que sean más bonitas, necesitaremos cambiar algunas de nuestras variables a sus datos etiquetados

```
tlaxt322%>%  
  dplyr::mutate(sex=sjlabelled::as_label(sex)) %>%  
  janitor::tabyl(sex)
```

	sex	n	percent
Hombre	5392	0.4767041	
Mujer	5919	0.5232959	

Para ver que esto es una distribución de frecuencias sería muy útil ver la proporción total, ello se realiza agregando un elemento más en nuestro código con una “tubería”:

```
tlaxt322%>%  
  mutate(sex=as_label(sex)) %>%  
  tabyl(sex) %>%  
  adorn_totals() #primer enchulamiento
```

	sex	n	percent
Hombre	5392	0.4767041	
Mujer	5919	0.5232959	
Total	11311	1.0000000	

Ahora, las proporciones son raras, y preferimos por los porcentajes.

```
tlaxt322%>%  
  mutate(sex=as_label(sex)) %>% # cambia los valores de la variable a sus etiquetas  
  tabyl(sex) %>% # para hacer la tabla  
  adorn_totals() %>% # añade totales  
  adorn_pct_formatting() # nos da porcentaje en lugar de proporción
```

	sex	n	percent
Hombre	5392	47.7%	
Mujer	5919	52.3%	
Total	11311	100.0%	

Vamos a darle una “ojeada” a esta variable

```
glimpse(tlaxt322$niv_ins)
```

```
dbl+lbl [1:11311] 4, 2, 3, 3, 3, 4, 2, 3, 4, 2, 2, 2, 2, 3, 3, 2, 3, 2, 2,...
@ label      : chr "Clasificación de la población ocupada por nivel de instrucción"
@ format.stata: chr "%12.0g"
@ labels      : Named num [1:6] 0 1 2 3 4 5
  ..- attr(*, "names")= chr [1:6] "No aplica" "Primaria incompleta" "Prrimaria completa" "S
```

Hoy hacemos la tabla, con las etiquetas:

```
tlaxt322%>%
  mutate(niv_ins=as_label(niv_ins)) %>% #esto sólo si hay etiquetas declaradas, recuerda
  tabyl(niv_ins)
```

	niv_ins	n	percent
	No aplica	714	0.0631243922
	Primaria incompleta	2173	0.1921138715
	Prrimaria completa	2025	0.1790292635
	Secundaria completa	3201	0.2829988507
	Medio superior y superior	3190	0.2820263460
	No especificado	8	0.0007072761

Para que no nos salgan las categorías sin datos podemos poner una opción dentro del comando “tabyl()”

```
tlaxt322%>%
  mutate(niv_ins=as_label(niv_ins)) %>%
  tabyl(niv_ins,
        show_missing_levels=F ) %>% # esta opción elimina los valores con 0
  adorn_totals()
```

	niv_ins	n	percent
	No aplica	714	0.0631243922
	Primaria incompleta	2173	0.1921138715
	Prrimaria completa	2025	0.1790292635
	Secundaria completa	3201	0.2829988507
	Medio superior y superior	3190	0.2820263460
	No especificado	8	0.0007072761
	Total	11311	1.0000000000

1.5.1 Cálculo de frecuencias

Las tablas de doble entrada tiene su nombre porque en las columnas entran los valores de una variable categórica, y en las filas de una segunda. Básicamente es como hacer un conteo de todas las combinaciones posibles entre los valores de una variable con la otra.

Por ejemplo, si quisiéramos combinar las dos variables que ya estudiamos lo podemos hacer, con una tabla de doble entrada:

```
tlaxt322%>%
  mutate(niv_ins=as_label(niv_ins)) %>% # para que las lea como factor
  mutate(sex=as_label(sex)) %>% # para que las lea como factor
  tabyl(niv_ins, sex, show_missing_levels=F ) %>% # incluimos aquí
  adorn_totals()
```

	niv_ins	Hombre	Mujer
	No aplica	389	325
	Primaria incompleta	1066	1107
	Primaria completa	950	1075
	Secundaria completa	1450	1751
	Medio superior y superior	1533	1657
	No especificado	4	4
	Total	5392	5919

Observamos que en cada celda confluyen los casos que comparten las mismas características:

```
tlaxt322%>%
  count(niv_ins==1 & sex==1) # nos da la segunda celda de la izquierda
```

```
# A tibble: 2 x 2
  `niv_ins == 1 & sex == 1`      n
  <lg1>                        <int>
1 FALSE                      10245
2 TRUE                        1066
```

1.5.2 Totales y porcentajes

De esta manera se colocan todos los datos. Si observa al poner la función “adorn_totals()” lo agregé como una nueva fila de totales, pero también podemos pedirle que agregue una columna de totales.

```
tlaxt322%>%
  mutate(niv_ins=as_label(niv_ins)) %>% # para que las lea como factor
  mutate(sex=as_label(sex)) %>% # para que las lea como factor
  tabyl(niv_ins, sex, show_missing_levels=F ) %>% # incluimos aquí sex
  adorn_totals("col")
```

	niv_ins	Hombre	Mujer	Total
	No aplica	389	325	714
	Primaria incompleta	1066	1107	2173
	Primaria completa	950	1075	2025
	Secundaria completa	1450	1751	3201
	Medio superior y superior	1533	1657	3190
	No especificado	4	4	8

O bien agregar los dos, introduciendo en el argumento `c("col", "row")` un vector de caracteres de las dos opciones requeridas:

```
tlaxt322%>%
  mutate(niv_ins=as_label(niv_ins)) %>% # para que las lea como factor
  mutate(sex=as_label(sex)) %>% # para que las lea como factor
  tabyl(niv_ins, sex, show_missing_levels=F ) %>% # incluimos aquí sexo
  adorn_totals(c("col", "row"))
```

	niv_ins	Hombre	Mujer	Total
	No aplica	389	325	714
	Primaria incompleta	1066	1107	2173
	Primaria completa	950	1075	2025
	Secundaria completa	1450	1751	3201
	Medio superior y superior	1533	1657	3190
	No especificado	4	4	8
	Total	5392	5919	11311

Del mismo modo, podemos calcular los porcentajes. Pero los podemos calcular de tres formas. Uno es que lo calculemos para los totales calculados para las filas, para las columnas o para el gran total poblacional.

Para columnas tenemos el siguiente código y los siguientes resultados:

```
tlaxt322%>%
  mutate(niv_ins=as_label(niv_ins)) %>% # para que las lea como factor
  mutate(sex=as_label(sex)) %>% # para que las lea como factor
  tabyl(niv_ins, sex, show_missing_levels=F ) %>% # incluimos aquí sexo
  adorn_totals(c("col", "row")) %>%
```

```
adorn_percentages("col") %>% # Divide los valores entre el total de la columna
adorn_pct_formatting() # lo vuelve porcentaje
```

	niv_ins	Hombre	Mujer	Total
	No aplica	7.2%	5.5%	6.3%
	Primaria incompleta	19.8%	18.7%	19.2%
	Pprimaria completa	17.6%	18.2%	17.9%
	Secundaria completa	26.9%	29.6%	28.3%
	Medio superior y superior	28.4%	28.0%	28.2%
	No especificado	0.1%	0.1%	0.1%
	Total	100.0%	100.0%	100.0%

Cuando se hagan cuadros de distribuciones (que todas sus partes suman 100), los porcentajes pueden ser una gran ayuda para la interpretación, sobre todos cuando se comparan poblaciones de categorías de diferente tamaño. Por lo general, queremos que los cuadros nos den información de donde están los totales y su 100%, de esta manera el lector se puede guiar de porcentaje con respecto a qué está leyendo. En este caso, vemos que el 100% es común en la última fila.

Veamos la diferencia de cómo podemos leer la misma celda, pero hoy, hemos calculado los porcentajes a nivel de fila:

```
tlaxt322%>%
mutate(niv_ins=as_label(niv_ins)) %>% # para que las lea como factor
mutate(sex=as_label(sex)) %>% # para que las lea como factor
tabyl(niv_ins, sex, show_missing_levels=F ) %>%
adorn_totals(c("col", "row")) %>%
adorn_percentages("row") %>% # Divide los valores entre el total de la fila
adorn_pct_formatting() # lo vuelve porcentaje
```

	niv_ins	Hombre	Mujer	Total
	No aplica	54.5%	45.5%	100.0%
	Primaria incompleta	49.1%	50.9%	100.0%
	Pprimaria completa	46.9%	53.1%	100.0%
	Secundaria completa	45.3%	54.7%	100.0%
	Medio superior y superior	48.1%	51.9%	100.0%
	No especificado	50.0%	50.0%	100.0%
	Total	47.7%	52.3%	100.0%

Finalmente, podemos calcular los porcentajes con referencia a la población total en análisis. Es decir la celda en la esquina inferior derecha de nuestra tabla original.

```

tlaxt322%>%
  mutate(niv_ins=as_label(niv_ins)) %>% # para que las lea como factor
  mutate(sex=as_label(sex)) %>% # para que las lea como factor
  tabyl(niv_ins, sex, show_missing_levels=F ) %>% # incluimos aquí sexo
  adorn_totals(c("col", "row")) %>%
  adorn_percentages("all") %>% # Divide los valores entre el total de la población
  adorn_pct_formatting() # lo vuelve porcentaje

```

	niv_ins	Hombre	Mujer	Total
	No aplica	3.4%	2.9%	6.3%
	Primaria incompleta	9.4%	9.8%	19.2%
	Pprimaria completa	8.4%	9.5%	17.9%
	Secundaria completa	12.8%	15.5%	28.3%
Medio superior y superior		13.6%	14.6%	28.2%
	No especificado	0.0%	0.0%	0.1%
	Total	47.7%	52.3%	100.0%

Chapter 2

Repaso de visualización de datos y {ggplot2}

2.1 Paquetes

```
if (!require("pacman")) install.packages("pacman")#instala pacman si se requiere
```

Loading required package: pacman

```
pacman::p_load(tidyverse,  
               readxl,  
               writexl,  
               haven,  
               sjlabelled,  
               janitor,  
               infer,  
               ggpubr,  
               magrittr,  
               gt,  
               GGally,  
               broom,  
               DescTools,  
               wesanderson)
```


2.2 Cargando los datos

Desde STATA

```
tlaxt322 <- read_dta("./datos/tlaxt322.dta") %>%  
  clean_names()
```

Desde Excel:

```
ICI_2021 <- read_excel("./datos/ICI_2021.xlsx",  
                      sheet = "para_importar") %>%  
  clean_names()
```

New names:

```
* `` -> `...2`
```

```
tlaxt322 %<>%  
  filter(r_def==0) %>%  
  filter(!c_res==2)
```

2.3 *Grammar of tables: gt*

Es un paquete que nos permite poner nuestras tablas en mejores formatos.

Guardemos un ejemplo anterior en un objeto

```
mi_tabla<-tlaxt322%>%  
  mutate(niv_ins=as_label(niv_ins)) %>% # para que las lea como factor  
  mutate(sex=as_label(sex)) %>% # para que las lea como factor  
  tabyl(niv_ins, sex, show_missing_levels=F ) %>% # incluimos aquí sexo  
  adorn_totals(c("col", "row")) %>%  
  adorn_percentages("all") %>% # Divide los valores entre el total de la población  
  adorn_pct_formatting() # lo vuelve porcentaje
```

Veamos qué pasa con el comando “gt”

```
gt_tabla<-gt(mi_tabla)  
gt_tabla
```

niv_ins	Hombre	Mujer	Total
---------	--------	-------	-------

No aplica	3.4%	2.9%	6.3%
Primaria incompleta	9.4%	9.8%	19.2%
Primeraria completa	8.4%	9.5%	17.9%
Secundaria completa	12.8%	15.5%	28.3%
Medio superior y superior	13.6%	14.6%	28.2%
No especificado	0.0%	0.0%	0.1%
Total	47.7%	52.3%	100.0%

Con este formato será bastante sencillo agregar títulos y demás:

```
gt_tabla<-gt_tabla %>%
  tab_header(
    title = "Distribución del sexo de la población según nivel de escolaridad",
    subtitle = "Tlaxcala, trimestre III de 2022"
  )

gt_tabla
```

Distribución del sexo de la población según nivel de escolaridad
Tlaxcala, trimestre III de 2022

niv_ins	Hombre	Mujer	Total
No aplica	3.4%	2.9%	6.3%
Primaria incompleta	9.4%	9.8%	19.2%
Primeraria completa	8.4%	9.5%	17.9%
Secundaria completa	12.8%	15.5%	28.3%
Medio superior y superior	13.6%	14.6%	28.2%
No especificado	0.0%	0.0%	0.1%
Total	47.7%	52.3%	100.0%

Agreguemos la fuente a nuestra tabla:

```
gt_tabla<-gt_tabla %>%
  tab_source_note(
    source_note = "Fuente: Cálculos propios con datos de INEGI"
  )

gt_tabla
```

Distribución del sexo de la población según nivel de escolaridad
Tlaxcala, trimestre III de 2022

niv_ins	Hombre	Mujer	Total
No aplica	3.4%	2.9%	6.3%
Primaria incompleta	9.4%	9.8%	19.2%
Primaria completa	8.4%	9.5%	17.9%
Secundaria completa	12.8%	15.5%	28.3%
Medio superior y superior	13.6%	14.6%	28.2%
No especificado	0.0%	0.0%	0.1%
Total	47.7%	52.3%	100.0%

Fuente: Cálculos propios con datos de INEGI

Checa más de este paquete por aquí <https://gt.rstudio.com/articles/intro-creating-gt-tables.html>

2.4 Descriptivos para variables cuantitativas

Vamos a empezar a revisar los gráficos para variables cuantitativas.

2.4.1 Medidas numéricas básicas

5 números

```
summary(tlaxt322$ing_x_hrs) ## ingreso por horas
```

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00   0.00   0.00  12.10  18.75 1356.59
```

Con pipes se pueden crear “indicadores” de nuestras variables es un tibble

```
tlaxt322 %>%
  summarise(nombre_indicador=mean(ing_x_hrs, na.rm=T))
```

```
# A tibble: 1 x 1
  nombre_indicador
      <dbl>
1             12.1
```

2.5 Visualización de datos, un pequeño disclaimer

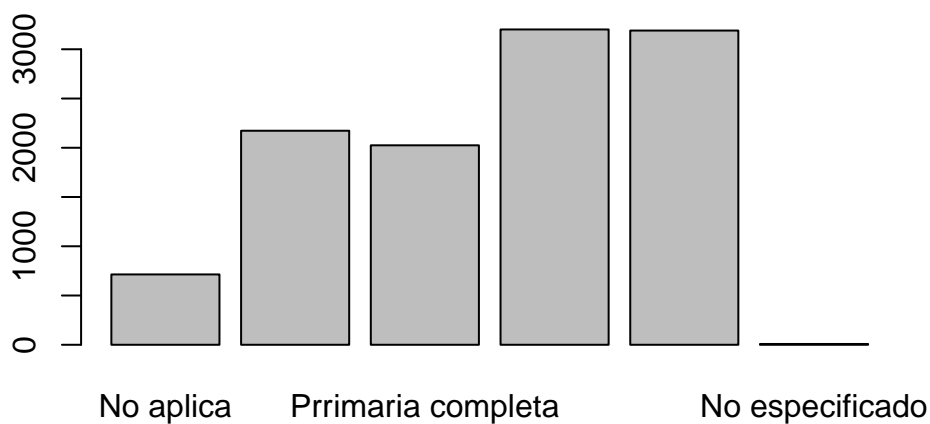
Hay cursos específicos de visualización de datos. Es maravilloso pero también requiere que estudiemos bien qué tipo de datos tenemos y cuáles son nuestros objetivos.

Me gusta mucho este recurso: <https://www.data-to-viz.com/>

2.5.1 Gráficos de base

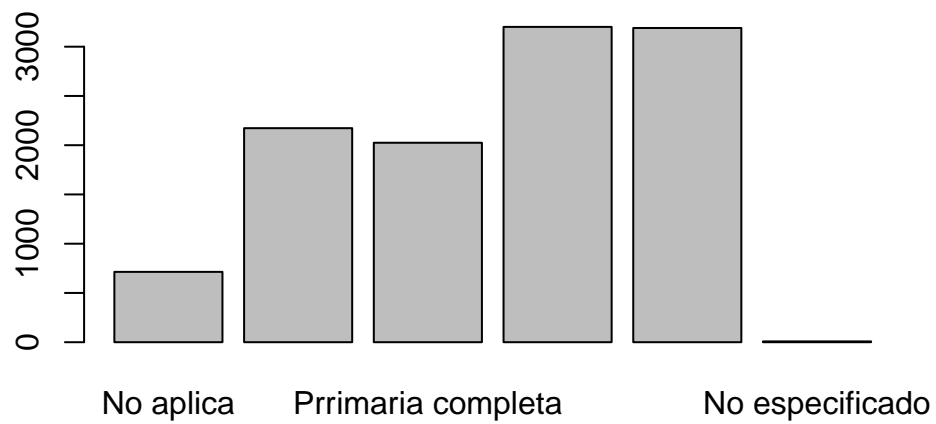
“plot()” Es la función más simple.

```
plot(as_label(tlaxt322$niv_ins))
```



Esto es igual que:

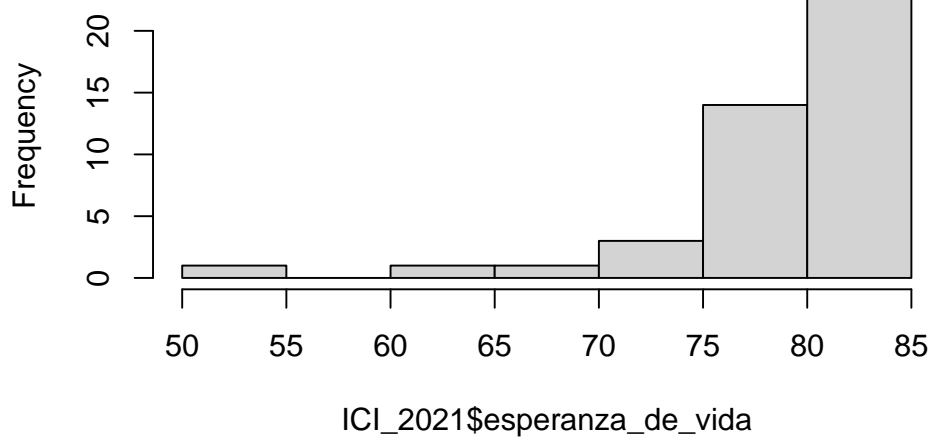
```
barplot(table(as_label(tlaxt322$niv_ins)))
```



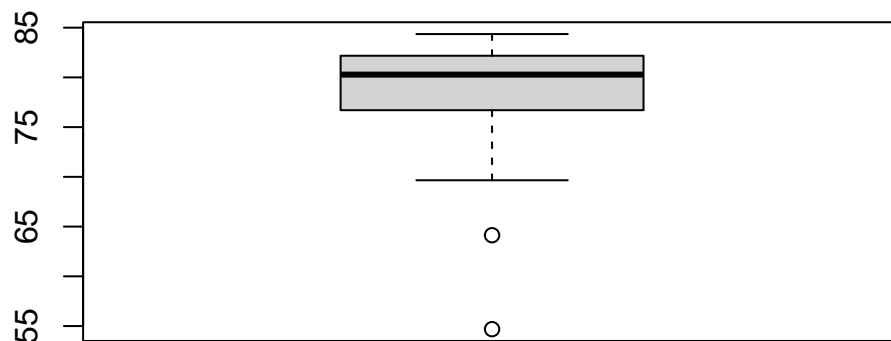
Histograma y el boxplot

```
hist(ICI_2021$esperanza_de_vida)
```

Histogram of ICI_2021\$esperanza_de_vida



```
boxplot(ICI_2021$esperanza_de_vida)
```



2.6 *Grammar of graphics: ggplot*

Hoy vamos a presentar a un gran paquete ¡Es de los famosos! Y tiene más de diez años.

- <https://qz.com/1007328/all-hail-ggplot2-the-code-powering-all-those-excellent-charts-is-10-years-old/>

“gg” proviene de “Grammar of Graphics”, funciona un poco como sintácticamente, de ahí su nombre.

Algunos recursos para aprender ggplot

- <https://ggplot2-book.org/> hecha por el mero mero.
- <http://sape.inf.usi.ch/quick-reference/ggplot2>
- <https://raw.githubusercontent.com/rstudio/cheatsheets/master/data-visualization-2.1.pdf>

Vamos a revisar una presentación que es muy interesante

- https://evamaerey.github.io/ggplot2_grammar_guide/ggplot2_grammar_guide.html
- <https://huygens.science.uva.nl/ggPlotteR/> Hace gráficos de ggplot con la base de datos de Gapminder

2.6.1 Un lienzo para dibujar

Para hacer un gráfico, ggplot2 tiene el comando “ggplot()”. Hacer gráficos con esta función tiene una lógica aditiva. Lo ideal es que iniciemos estableciendo el mapeo estético de nuestro gráfico, con el comando `aes()`

```
g1<-tlaxt322 %>%
  ggplot(aes(as_label(niv_ins)))

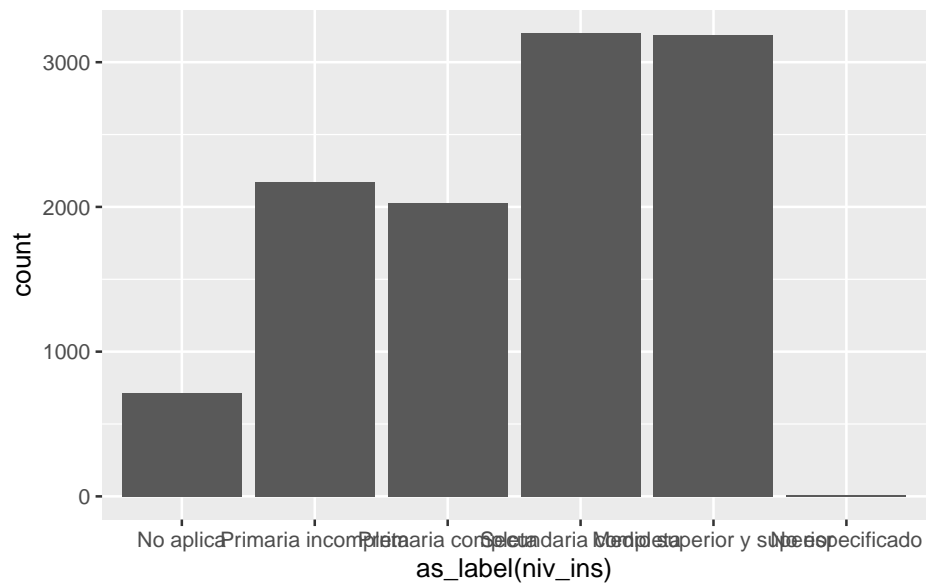
g1 ## imprime el lienzo
```



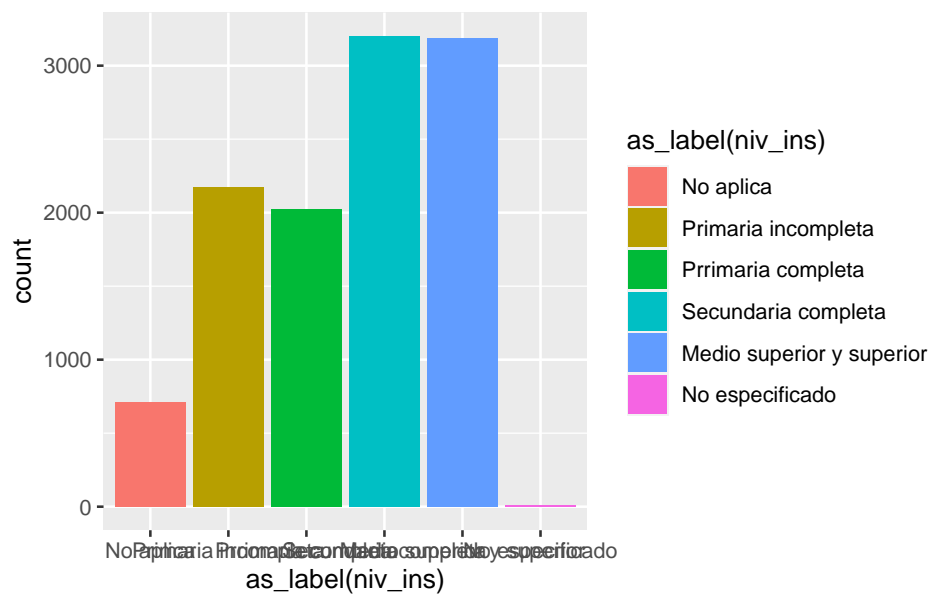
2.6.2 Gráficos univariados

2.6.2.1 Para cualitativas

```
g1 + geom_bar()
```

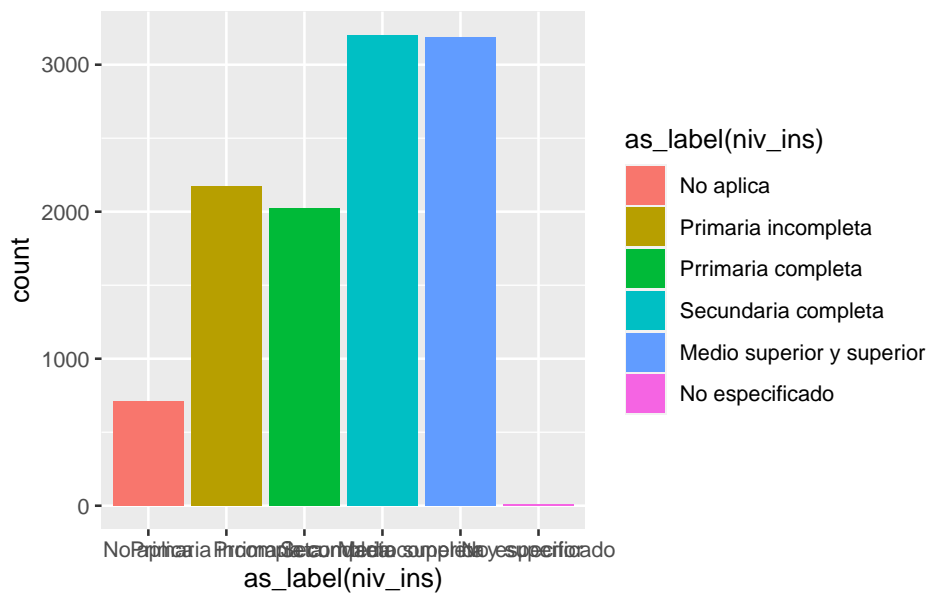


```
g1 + geom_bar(aes(
  fill = as_label(niv_ins)
)) ## colorea la geometría
```




```
## Esto es equivalente

tlaxt322 %>%
  ggplot(aes(as_label(niv_ins),
              fill = as_label(niv_ins)
            )
        ) + geom_bar()
```

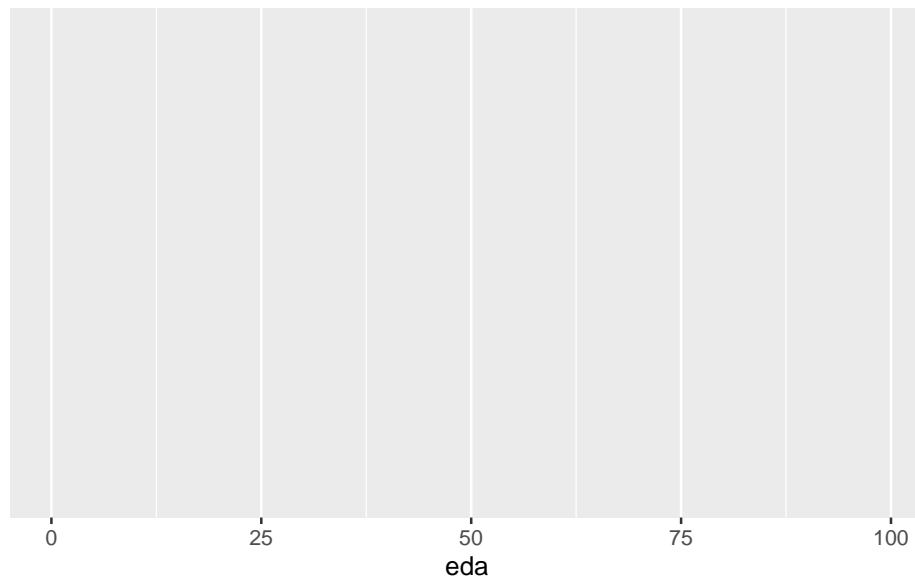


2.6.2.2 Para variables cuantitativas

Podemos hacer histogramas y gráficos de densidad, de manera fácil. La idea es agregar en nuestro “lienzo” una geometría, un valor para dibujar en él. Esto se agrega con un “+” y con la figura que se añadirá a nuestro gráfico.

```
g2<-tlaxt322 %>%
  ggplot(aes(eda))

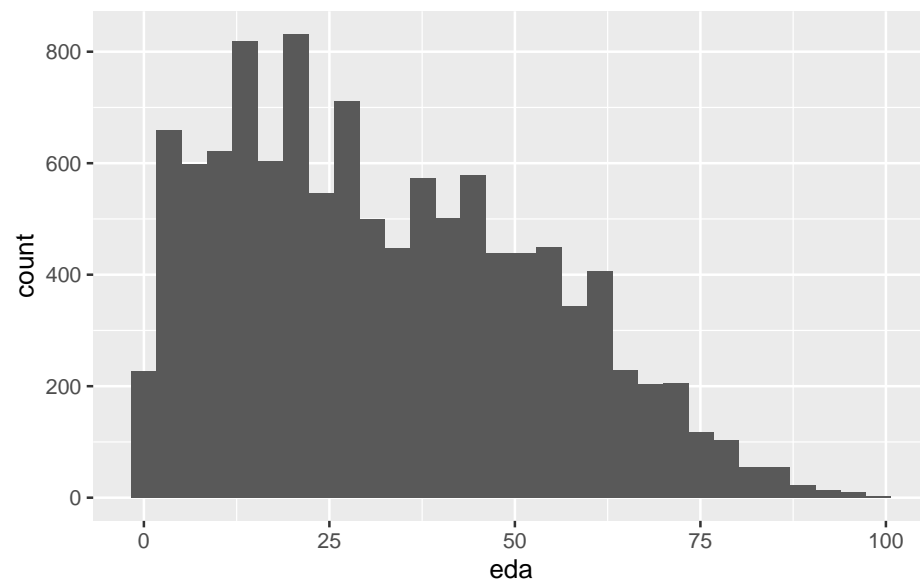
g2 ## imprime el lienzo
```



2.6.2.2.1 Histograma

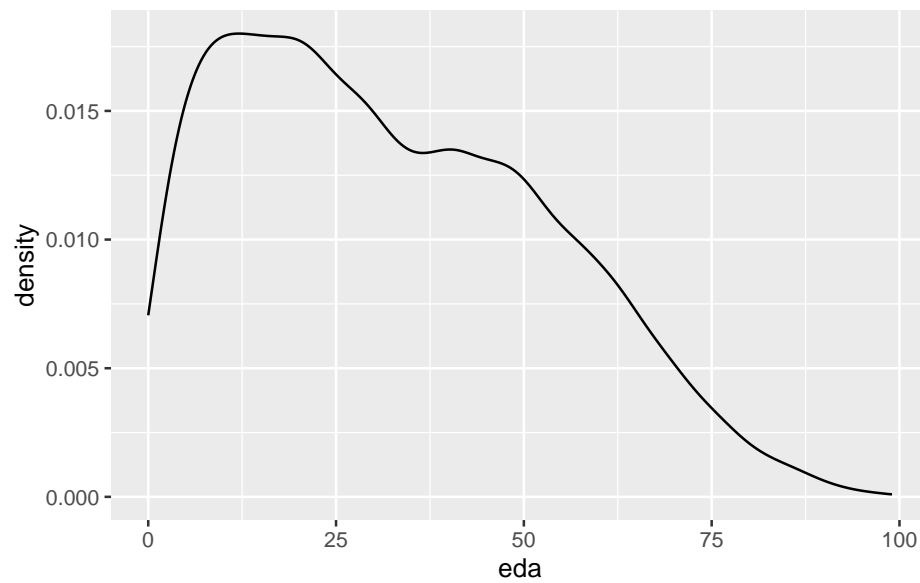
```
g2 + geom_histogram()
```

``stat_bin()`` using ``bins = 30``. Pick better value with ``binwidth``.



2.6.2.2.2 Densidad

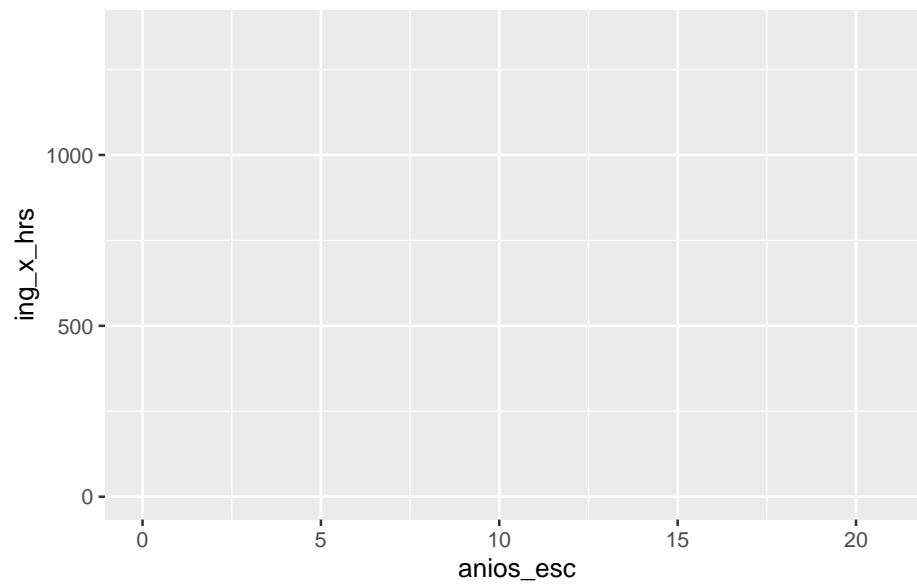
```
g2 + geom_density()
```



2.7 Intro a dos variables

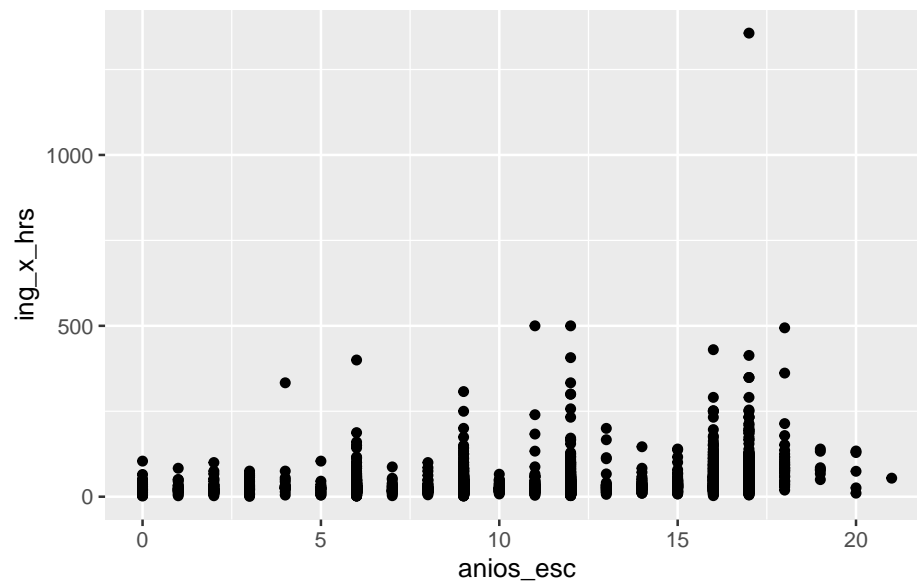
Para hacer un lienzo con dos variables:

```
tlaxt322 %>%  
  filter(ing_x_hrs>0) %>%  
  filter(anios_esc<99) %>% #  
  ggplot()+  
  aes(x = anios_esc,  
       y = ing_x_hrs)
```



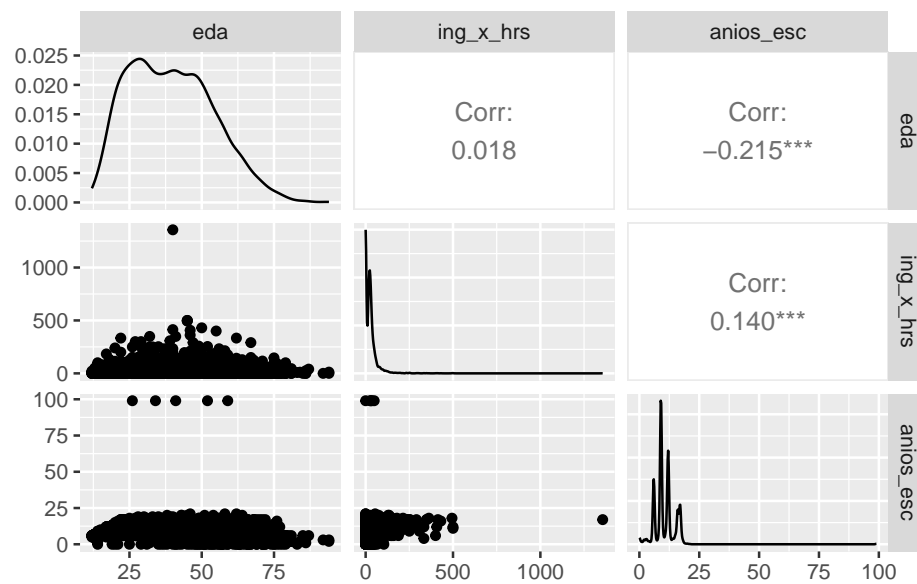
Con esto podemos agregar la geometría de puntito:

```
tlaxt322 %>%  
  filter(ing_x_hrs>0) %>%  
  filter(anios_esc<99) %>% #  
  ggplot()+  
  aes(x = anios_esc,  
       y = ing_x_hrs) +  
  geom_point()
```



Vamos a terminar con un código que resume mucho de lo que hemos visto hoy:

```
tlaxt322 %>%
  filter(clase2==1) %>% ## nos quedamos sólo con los ocupados
  select(eda, ing_x_hrs, anios_esc) %>%
  GGally::ggpairs()
```



Chapter 3

Introducción a la inferencia

3.1 Paquetes

```
if (!require("pacman")) install.packages("pacman")#instala pacman si se requiere
```

Loading required package: pacman

```
pacman::p_load(tidyverse,  
               readxl,  
               writexl,  
               haven,  
               sjlabelled,  
               janitor,  
               infer,  
               ggpubr,  
               magrittr,  
               gt,  
               GGally,  
               broom,  
               DescTools,  
               wesanderson,  
               gtsummary,  
               srvyr,  
               car)
```

3.2 Cargando los datos

Desde STATA

```
tlaxt322 <- read_dta("./datos/tlaxt322.dta") %>%  
  clean_names() %>%  
  filter(r_def==0) %>%  
  filter(!c_res==2)
```

Desde Excel:

```
ICI_2021 <- read_excel("./datos/ICI_2021.xlsx",  
                        sheet = "para_importar") %>%  
  clean_names()
```

New names:

```
* `` -> `...2`
```

3.3 Hipótesis e intervalos de confianza

3.3.1 t-test

Este comando nos sirve para calcular diferentes tipos de test, que tienen como base la distribución t

Univariado para estimación

```
t.test(tlaxt322$ing_x_hrs) # pero no tenemos los filtro
```

One Sample t-test

```
data: tlaxt322$ing_x_hrs  
t = 42.362, df = 11310, p-value < 2.2e-16  
alternative hypothesis: true mean is not equal to 0  
95 percent confidence interval:  
 11.53964 12.65937  
sample estimates:  
mean of x  
 12.09951
```

Un truco para poder utilizar funciones de base con formato *tidy*

```
tlaxt322 %>%
  filter(clase2==1) %>% #Filtro de ocupados
  filter(ing_x_hrs>0) %>% #Filtros de quienes reportaron ingresos
  with(
    t.test(ing_x_hrs)
  )
```

One Sample t-test

```
data: ing_x_hrs
t = 51.961, df = 3631, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 36.25925 39.10282
sample estimates:
mean of x
 37.68103
```

Vamos a quedarnos a con esta población objetivo:

Univariado para hipótesis específica

$$H_o : \mu = 40$$

$$H_{a1} : \mu < 40$$

$$H_{a2} : \mu \neq 40$$

$$H_{a3} : \mu > 40$$

Si hacemos explícita la H_0

```
tlaxt322 %>%
  filter(clase2==1) %>% #Filtro de ocupados
  filter(ing_x_hrs>0) %>% #Filtros de quienes reportaron ingresos
  with(
    t.test(tlaxt322$ing_x_hrs, mu=40)
  )
```

One Sample t-test

```
data: tlaxt322$ing_x_hrs
t = -97.684, df = 11310, p-value < 2.2e-16
```



```

alternative hypothesis: true mean is not equal to 40
95 percent confidence interval:
 11.53964 12.65937
sample estimates:
mean of x
 12.09951

```

Para hacer explícitas las hipótesis alternativas

```

tlaxt322 %>%
  filter(clase2==1) %>% #Filtro de ocupados
  filter(ing_x_hrs>0) %>% #Filtros de quienes reportaron ingresos
  with(
    t.test(tlaxt322$ing_x_hrs, mu=40, alternative = "two.sided") #default y de dos colas
  )

```

One Sample t-test

```

data: tlaxt322$ing_x_hrs
t = -97.684, df = 11310, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 40
95 percent confidence interval:
 11.53964 12.65937
sample estimates:
mean of x
 12.09951

```

```

tlaxt322 %>%
  filter(clase2==1) %>% #Filtro de ocupados
  filter(ing_x_hrs>0) %>% #Filtros de quienes reportaron ingresos
  with(
    t.test(tlaxt322$ing_x_hrs, mu=40, alternative = "greater") # cola derecha
  )

```

One Sample t-test

```

data: tlaxt322$ing_x_hrs
t = -97.684, df = 11310, p-value = 1
alternative hypothesis: true mean is greater than 40
95 percent confidence interval:
 11.62966      Inf

```

sample estimates:

mean of x

12.09951

```
tlaxt322 %>%
  filter(clase2==1) %>% #Filtro de ocupados
  filter(ing_x_hrs>0) %>% #Filtros de quienes reportaron ingresos
  with(
    t.test(tlaxt322$ing_x_hrs, mu=40, alternative = "less") # cola izquierda
  )
```

One Sample t-test

data: tlaxt322\$ing_x_hrs

t = -97.684, df = 11310, p-value < 2.2e-16

alternative hypothesis: true mean is less than 40

95 percent confidence interval:

-Inf 12.56935

sample estimates:

mean of x

12.09951

3.3.2 Enchulando un poquito

Los resultados tienen la info, pero la podemos almacenar en un objeto. Con los cálculos de modelos es muy útil guardarlos para compararlos.

```
t.test0<-tlaxt322 %>%
  filter(clase2==1) %>% #Filtro de ocupados
  filter(ing_x_hrs>0) %>% #Filtros de quienes reportaron ingresos
  with(
    t.test(tlaxt322$ing_x_hrs, mu=40)
  )
```

Veamos si lo imprimimos

```
t.test0
```

One Sample t-test

data: tlaxt322\$ing_x_hrs

```
t = -97.684, df = 11310, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 40
95 percent confidence interval:
 11.53964 12.65937
sample estimates:
mean of x
 12.09951
```

```
broom::tidy(t.test0)
```

```
# A tibble: 1 x 8
  estimate statistic p.value parameter conf.low conf.high method      alter~1
  <dbl>      <dbl>   <dbl>      <dbl>    <dbl>    <dbl> <chr>      <chr>
1    12.1      -97.7     0      11310     11.5     12.7 One Sample t-- two.si~
# ... with abbreviated variable name 1: alternative
```

La función “tidy()” hace que el resultado se vuelva un “tibble”, una tabla muy compatible con el tidyverse. Esto puede ser útil cuando queremos ir comparando estimaciones.

Anteriormente vimos con base cómo hacer inferencia. El paquete “infer” tiene también elementos para inferencia, pero en formato más compatible con tidyverse.

```
tlaxt322 %>%
  filter(clase2==1) %>% #Filtro de ocupados
  filter(ing_x_hrs>0) %>% #Filtros de quienes reportaron ingresos
  infer::t_test(response = ing_x_hrs, mu = 40)
```

```
# A tibble: 1 x 7
  statistic t_df p_value alternative estimate lower_ci upper_ci
  <dbl> <dbl>   <dbl> <chr>      <dbl>    <dbl>    <dbl>
1    -3.20  3631 0.00140 two.sided     37.7     36.3     39.1
```

Como vemos nos da el mismo resultado anterior, pero nos da directamente el resultado en formato tidy.

Si solo queremos el estimador de “t”

```
tlaxt322 %>%
  t_stat(response = ing_x_hrs, mu = 40)
```

Warning: The t_stat() wrapper has been deprecated in favor of the more general observe(). Please use that function instead.

t
-97.68389

Más de este paquete <https://infer.netlify.app/>

3.3.3 Prueba para proporción

Vamos a revisar la proporción de hombres y mujeres en términos de participación laboral.

El comando de base es menos flexible:

```
prop<-table(tlaxt322[tlaxt322$clase1>0,]$clase1)
prop.test(prop)
```

1-sample proportions test with continuity correction

```
data: prop, null probability 0.5
X-squared = 259.77, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.5739084 0.5941550
sample estimates:
      p
0.5840669
```

Los filtros se complican un poco...

```
tlaxt322 %>%
  filter(eda>14 & eda<99) %>%
  mutate(clase1=as_label(clase1)) %>% #oo
  tabyl(clase1)
```

	clase1	n	percent
	No aplica	0	0.0000000
	Población económicamente activa	5326	0.6208183
	Población no económicamente activa	3253	0.3791817

Vamos a aprovechar para re-etiquetar la variable clase1

```
etiqueta_pea<-c("PEA", "PNEA") # un vector con las etiquetas
```

```
tlaxt322 %>%
  filter(eda>14 & eda<99) %>%
  sjlabelled::set_labels(clase1, labels=etiqueta_pea) %>%
  mutate(clase1=as_label(clase1)) %>%
  tabyl(clase1)
```

```
clase1    n  percent
PEA  5326 0.6208183
PNEA 3253 0.3791817
```

En formato tidy

```
tlaxt322 %>%
  filter(eda>14 & eda<99) %>%
  with(
    table(clase1)
  ) %>%
  prop.test()
```

1-sample proportions test with continuity correction

```
data:  ., null probability 0.5
X-squared = 500.43, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.6104410 0.6310868
sample estimates:
      p
0.6208183
```

En base necesita que se alimente de un objeto tipo table, el cual es menos manejable. Por eso utilizaremos más el paquete `{infer}`

```
tlaxt322 %>%
  filter(eda>14 & eda<99) %>%
  set_labels(clase1, labels=etiqueta_pea) %>%
  mutate(clase1=as_label(clase1)) %>%
  infer::prop_test(clase1 ~ NULL ,
    p=0.7,
    alternative="less")
```

```
# A tibble: 1 x 4
  statistic chisq_df p_value alternative
    <dbl>      <int>    <dbl> <chr>
1      256.         1 7.22e-58 less

# Para que nos dé Z
tlaxt322 %>%
  filter(eda>14 & eda<99) %>%
  set_labels(clase1, labels=etiqueta_pea) %>%
  mutate(clase1=as_label(clase1)) %>%
  infer::prop_test(clase1 ~ NULL ,
    p=0.7,
    alternative="less",
    success = "PEA", # necesitamos establecer el éxito
    z=TRUE)

# A tibble: 1 x 3
  statistic p_value alternative
    <dbl>      <dbl> <chr>
1     -16.0 5.97e-58 less
```

3.4 Análisis de varianza

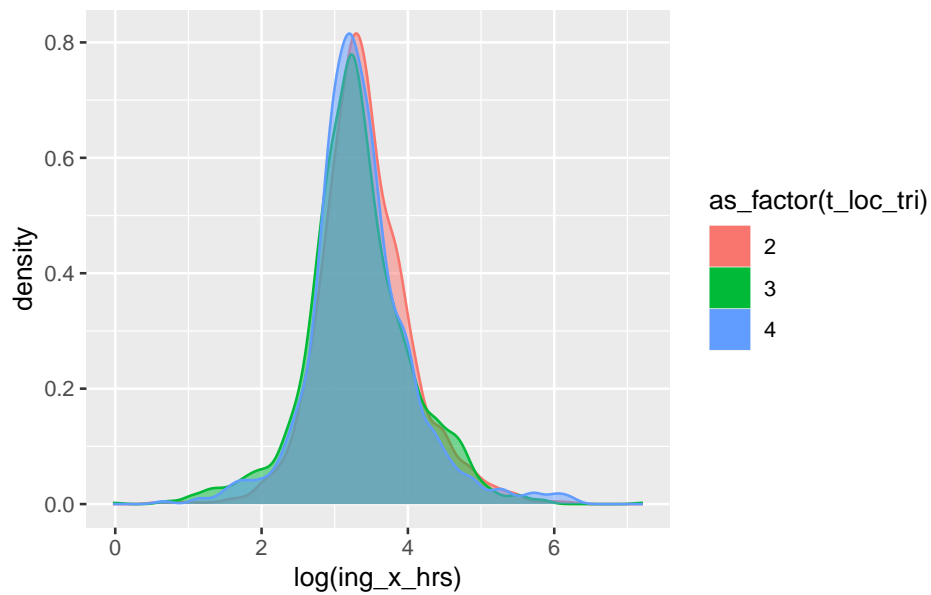
Análisis de varianza. Haremos la versión más simple. Para ver el efecto de un factor sobre una variable cualitativa (oneway). Revisaremos si la región de residencia de los trabajadores tiene un efecto en la distribución de los ingresos por trabajo.

3.4.1 Primero un gráfico

la ANOVA se basa en que nuestra variable es normal. Quitaremos los outliers

```
lienzo_bi <-tlaxt322 %>%
  filter(clase2==1 & !ing_x_hrs==0) %>%
  ggplot(aes(x=log(ing_x_hrs), fill=as_factor(t_loc_tri),
    color=as_factor(t_loc_tri),
    alpha=I(0.5)))

lienzo_bi + geom_density()
```



La prueba ANOVA o análisis de varianza, nos dice cuánto de nuestra variable se ve explicado por un factor.

$$H_o : \mu_1 = \mu_2 = \mu_3 = \mu_4$$

H_a : alguna de las medias es diferente

En los modelos es muy útil guardar nuestros resultados como un objeto

```
anova<-tlaxt322 %>%
  filter(clase2==1) %>%
  with(aov(ing_x_hrs ~ as_factor(t_loc_tri)))

summary(anova)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
as_factor(t_loc_tri)	2	5837	2918	1.783	0.168
Residuals	5171	8466230	1637		

Con tidy:

```
tidy(anova)
```

```
# A tibble: 2 x 6
  term          df      sumsq meansq statistic p.value
<chr>      <dbl>    <dbl>  <dbl>    <dbl>    <dbl>
1 as_factor(t_loc_tri)      2    5837.   2919.      1.78    0.168
2 Residuals          5171 8466230.   1637.      NA      NA
```

3.4.2 Comparación entre grupos

¿si es significativo cuáles diferencias entre los grupos lo son?

```
TukeyHSD(anova)
```

```
Tukey multiple comparisons of means
95% family-wise confidence level
```

```
Fit: aov(formula = ing_x_hrs ~ as_factor(t_loc_tri))
```

```
$`as_factor(t_loc_tri)`
      diff      lwr      upr      p adj
3-2 -0.583763 -3.5032238 2.335698 0.8859508
4-2  2.369155 -1.4109047 6.149215 0.3058249
4-3  2.952918 -0.7414905 6.647326 0.1463950
```

3.4.3 Supuestos de ANOVA

- Las observaciones se obtienen de forma independiente y aleatoria de la población definida por los niveles del factor
- Los datos de cada nivel de factor se distribuyen normalmente.
- Estas poblaciones normales tienen una varianza común.

```
#Prueba Bartlett para ver si las varianzas son iguales
```

```
tlaxt322 %>%
  filter(clase2==1) %>%
  with(bartlett.test(ing_x_hrs ~ as_factor(t_loc_tri)))
```

```
Bartlett test of homogeneity of variances
```

```
data: ing_x_hrs by as_factor(t_loc_tri)
Bartlett's K-squared = 132, df = 2, p-value < 2.2e-16
```


La prueba tiene una Ho “Las varianzas son iguales”

```
#Test Normalidad
tlaxt322 %>%
  filter(clase2==1) %>%
  filter(ing_x_hrs>0) %>%
  with(
    ks.test(ing_x_hrs, "pnorm", mean=mean(ing_x_hrs), sd=sd(ing_x_hrs))
  )
```

Warning in ks.test.default(ing_x_hrs, "pnorm", mean = mean(ing_x_hrs), sd = sd(ing_x_hrs)): ties should not be present for the Kolmogorov-Smirnov test

Asymptotic one-sample Kolmogorov-Smirnov test

```
data: ing_x_hrs
D = 0.23129, p-value < 2.2e-16
alternative hypothesis: two-sided
```

La prueba tiene una Ho “La variable es normal”

¿Qué hacer?

3.4.4 Kruskal-Wallis test

Hay una prueba muy parecida que se basa en el orden de las observaciones, y se lee muy parecida a la ANOVA

```
kruskal<-tlaxt322 %>%
  filter(clase2==1) %>%
  with(
    kruskal.test(ing_x_hrs ~ as_factor(t_loc_tri))
  )

kruskal
```

Kruskal-Wallis rank sum test

```
data: ing_x_hrs by as_factor(t_loc_tri)
Kruskal-Wallis chi-squared = 6.7713, df = 2, p-value = 0.03385
```

Para ver las comparaciones tenemos que usar el `DunnTest()`, del paquete `{DescTools}`

```
tlaxt322 %>%
  filter(clase2==1) %>%
  with(
    DescTools::DunnTest(ing_x_hrs ~ as_factor(t_loc_tri))
  )
```

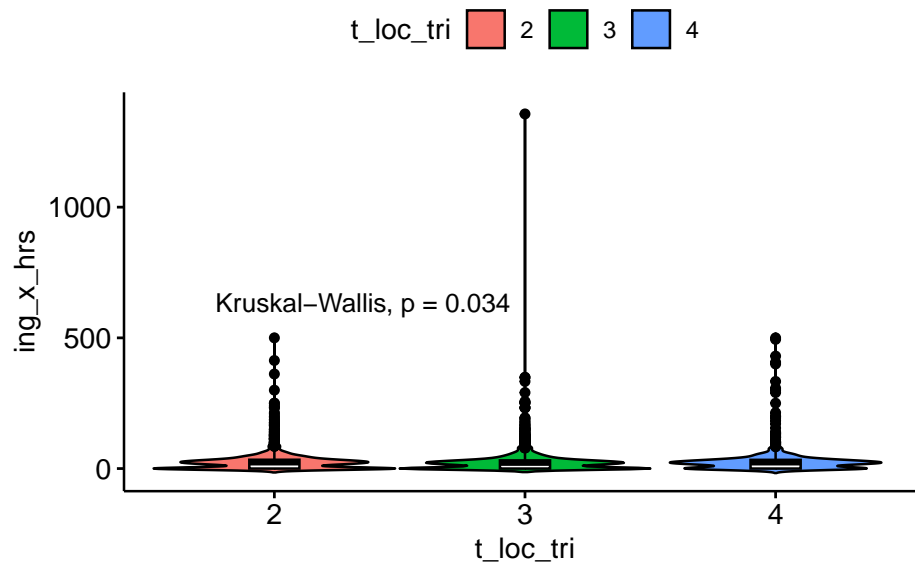
Dunn's test of multiple comparisons using rank sums : holm

```
      mean.rank.diff    pval
3-2      -83.88836 0.1288
4-2       52.69560 0.3695
4-3      136.58397 0.0520 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.4.4.1 Un gráfico coqueto:

Se hace con `ggpubr`

```
tlaxt322 %>%
  filter(clase2==1) %>%
  ggpubr::ggviolin(x = "t_loc_tri", y = "ing_x_hrs", fill = "t_loc_tri",
    add = "boxplot", add.params = list(fill = "white")) +
  stat_compare_means(label.y = 600) # Add the p-value
```



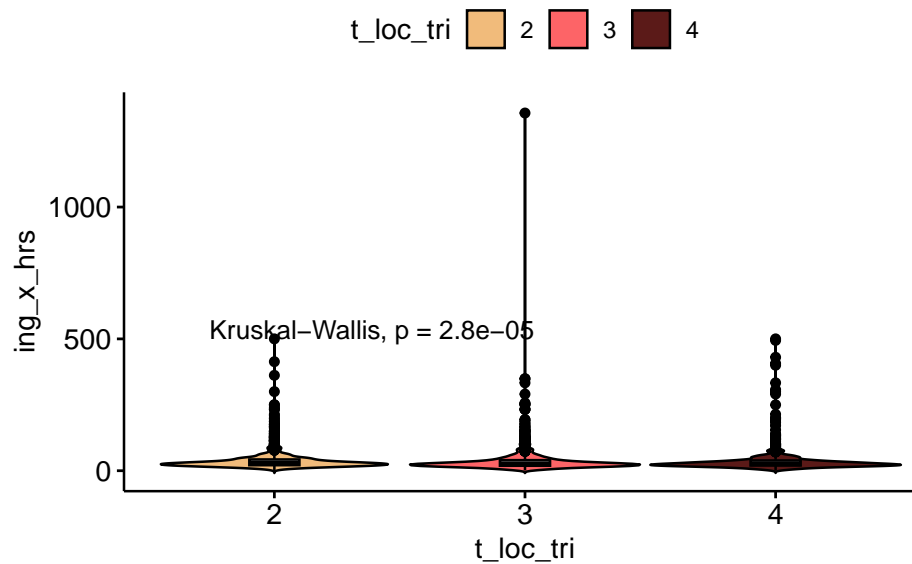
```
comparaciones <- list( c("1", "2"), c("2", "3"), c("1", "3"),
                      c("1", "4"), c("2", "4"), c("3", "4") )
```

```
#Un gráfiquito accesorio 2:
```

```
tlaxt322 %>%
  filter(clase2==1) %>%
  filter(ing_x_hrs>0) %>%
  ggpubr::ggviolin(x = "t_loc_tri", y = "ing_x_hrs", fill = "t_loc_tri",
                  palette = wesanderson::wes_palette("GrandBudapest1", 4, type="discrete"),
                  add = "boxplot", add.params = list(fill = "white"))+
  stat_compare_means(comparisons = comparaciones, label = "p.signif")+ # Add significance
  stat_compare_means(label.y = 500) # Add global the p-value
```

```
[1] FALSE
```

```
Warning: Computation failed in `stat_signif()`
Caused by error in `if (scales$x$map(comp[1]) == data$group[1] | manual) ...`:
! valor ausente donde TRUE/FALSE es necesario
```



3.5 Factores de expansión

3.5.1 La función tally

El comando `taby1()` del paquete `{janitor}` es muy útil pero no es compatible con los factores de expansión. En realidad, `taby1()` nos ahorra un poco el hecho de tener que agrupar nuestra base en categorías y luego hacer un conteo para cada una de ellas. `tally()` es un comando que nos hace ese conteo y `group_by()` nos agrupa las observaciones de nuestra base de datos para hacer cualquier operación.

```
tlaxt322 %>%
  group_by(as_label(sex)) %>%
  tally(fac_tri) %>% #nombre del factor
  adorn_totals() # Agrega total
```

```
as_label(sex)      n
Hombre  650712
Mujer    720359
Total  1371071
```

Podemos usar funciones de `adorn_...` de `{janitor}`

```
tlaxt322 %>%
  group_by(as_label(sex)) %>%
  tally(fac_tri) %>% #nombre del factor
  adorn_totals() %>% # Agrega total
  adorn_percentages("all") %>%
  adorn_pct_formatting()
```

```
as_label(sex)      n
      Hombre 47.5%
      Mujer  52.5%
      Total 100.0%
```

3.6 Otras formas

La función `count()` también permite dar pesos

```
tlaxt322 %>%
  dplyr::count(sex, niv_ins, wt = fac_tri)
```

```
# A tibble: 12 x 3
   sex      niv_ins      n
  <dbl+lbl> <dbl+lbl> <dbl>
1 1 [Hombre] 0 [No aplica] 47513
2 1 [Hombre] 1 [Primaria incompleta] 130266
3 1 [Hombre] 2 [Pprimaria completa] 116761
4 1 [Hombre] 3 [Secundaria completa] 178274
5 1 [Hombre] 4 [Medio superior y superior] 177459
6 1 [Hombre] 5 [No especificado] 439
7 2 [Mujer] 0 [No aplica] 40662
8 2 [Mujer] 1 [Primaria incompleta] 137557
9 2 [Mujer] 2 [Pprimaria completa] 132531
10 2 [Mujer] 3 [Secundaria completa] 213295
11 2 [Mujer] 4 [Medio superior y superior] 195934
12 2 [Mujer] 5 [No especificado] 380
```

Es compatible con etiquetas

```
tlaxt322 %>%
  count(as_label(sex), as_label(niv_ins), wt = fac_tri)
```

```
# A tibble: 12 x 3
  `as_label(sex)` `as_label(niv_ins)`      n
  <fct>          <fct>          <dbl>
1 Hombre        No aplica        47513
2 Hombre        Primaria incompleta 130266
3 Hombre        Pprimaria completa 116761
4 Hombre        Secundaria completa 178274
5 Hombre        Medio superior y superior 177459
6 Hombre        No especificado    439
7 Mujer         No aplica        40662
8 Mujer         Primaria incompleta 137557
9 Mujer         Pprimaria completa 132531
10 Mujer        Secundaria completa 213295
11 Mujer        Medio superior y superior 195934
12 Mujer        No especificado    380
```

Podemos mover un poquito con `pivot_wider` para que se vea más a lo que acostumbramos a una tabla de frecuencias

```
tlaxt322 %>%
  mutate_at(vars(sex, niv_ins), as_label) %>%
  count(sex, niv_ins, wt = fac_tri) %>%
  tidyr::pivot_wider(names_from = sex,
                    values_from = n)
```

```
# A tibble: 6 x 3
  niv_ins      Hombre  Mujer
  <fct>          <dbl> <dbl>
1 No aplica      47513  40662
2 Primaria incompleta 130266 137557
3 Pprimaria completa 116761 132531
4 Secundaria completa 178274 213295
5 Medio superior y superior 177459 195934
6 No especificado    439    380
```

```
tlaxt322 %>%
  mutate_at(vars(sex, niv_ins), as_label) %>% # otra forma de mutate y as_label
  count(sex, niv_ins, wt = fac_tri) %>%
  pivot_wider(names_from = sex,
              values_from = n) %>%
  adorn_totals() %>% # Agrega total
  adorn_percentages("col") %>%
  adorn_pct_formatting()
```

	niv_ins	Hombre	Mujer
	No aplica	7.3%	5.6%
	Primaria incompleta	20.0%	19.1%
	Primaria completa	17.9%	18.4%
	Secundaria completa	27.4%	29.6%
	Medio superior y superior	27.3%	27.2%
	No especificado	0.1%	0.1%
	Total	100.0%	100.0%

#Diseño complejo

Hay muchos diseños muestrales, asumiremos el diseño simple, pero hay que revisar la documentación de la base

```
# Muestreo aleatorio
tlax_srvy <- tlaxt322 %>%
  as_survey_design(weights = fac_tri)
```

Si revisamos las encuestas tiene un diseño complejo, hay estratos y unidades primarias de muestreo

```
# Muestreo estratificado
tlax_srvy <- tlaxt322 %>%
  as_survey_design(
    upm,
    strata = est_d_tri,
    weights = fac_tri,
    nest = TRUE)
```

Como vemos esto es un archivo bien grande, por lo que mejor vamos a seleccionar un par de variables:

```
# simple random sample
tlax_srvy <- tlaxt322 %>%
  select(upm, est_d_tri, fac_tri, starts_with("clase"),
    sex, eda, anios_esc, ing_x_hrs, fac_tri) %>%
  as_survey_design(
    upm,
    strata = est_d_tri,
    weights = fac_tri,
    nest = TRUE)
```

Para una media ponderada

```

tlax_srvy %>%
  filter(eda>14 & eda<99) %>% #filtro de edad para tabulados
  filter(clase2==1) %>% # sólo ocupados
  filter(ing_x_hrs>0) %>% # sólo con ingresos
  summarise(
    media_ponderada = survey_mean(ing_x_hrs, na.rm=T))

# A tibble: 1 x 2
  media_ponderada media_ponderada_se
      <dbl>          <dbl>
1      37.1          0.818

```

Este valor coincide con los datos publicados por INEGI, incluso el error estándar. Si queremos los intervalos de confianza:

```

tlax_srvy %>%
  filter(eda>14 & eda<99) %>% #filtro de edad para tabulados
  filter(clase2==1) %>% # sólo ocupados
  filter(ing_x_hrs>0) %>% # sólo con ingresos
  summarize(
    media_ponderada = survey_mean(ing_x_hrs,
                                   vartype = "ci") )

# A tibble: 1 x 3
  media_ponderada media_ponderada_low media_ponderada_upp
      <dbl>          <dbl>          <dbl>
1      37.1          35.5          38.7

```

```

tlax_srvy %>%
  filter(eda>14 & eda<99) %>% #filtro de edad para tabulados
  filter(clase2==1) %>% # sólo ocupados
  filter(ing_x_hrs>0) %>% # sólo con ingresos
  summarize(
    mediana_ponderada = survey_median(ing_x_hrs,
                                       vartype = "ci") )

# A tibble: 1 x 3
  mediana_ponderada mediana_ponderada_low mediana_ponderada_upp
      <dbl>          <dbl>          <dbl>
1      27.1          26.7          28

```



```

tlax_srvy %>%
  mutate(sex=as_label(sex)) %>%
  group_by(sex) %>% #variables cuali
  summarize(proportion = survey_mean(), # proporción
            total = survey_total() ) # totales

```

A tibble: 2 x 5

	sex	proportion	proportion_se	total	total_se
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>
1	Hombre	0.475	0.00397	650712	12123.
2	Mujer	0.525	0.00397	720359	12890.

Material anexo

Lista general de YouTube

https://www.youtube.com/playlist?list=PLDnSa5YhrAVlczarJEzgR_bxFGbg9MGEU

Scripts

Los scripts están en esta carpeta: https://github.com/aniuxa/posgrado_modelo3/tree/main/scripts

Cuestionarios

Primer cuestionario

Después de la sesión 1 y 2

[Cuestionario](#)

Sesión 1

Video

<https://youtu.be/D4KgiEX7rB8>

Cheat Sheets

- ggplot2: <https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf>