03

TALLER DE INVESTIGACIÓN APLICADA

Producción y análisis de datos

La importancia del workflow

Mauricio Bucca · Sociología UC · DobleA

Clase 3 · Mayo 2026 · Lunes 19 · 6–9 PM

HOY

El problema, la solución, el objetivo

*Reproducir todos los resultados con **un solo clic** — o una línea de código.*

BLOQUE 1

Diagnóstico

Cómo se ve un proyecto mal organizado y por qué nos cuesta tiempo, errores y credibilidad.

BLOQUE 2

Principios y herramientas

Qué es un workflow y por qué importa más que la herramienta; cada herramienta resuelve un problema concreto.

BLOQUE 3

Todo junto

Un caso completo de principio a fin — recapitulación con un ejemplo aplicado.

** Habrá 2 breaks de 10 min*

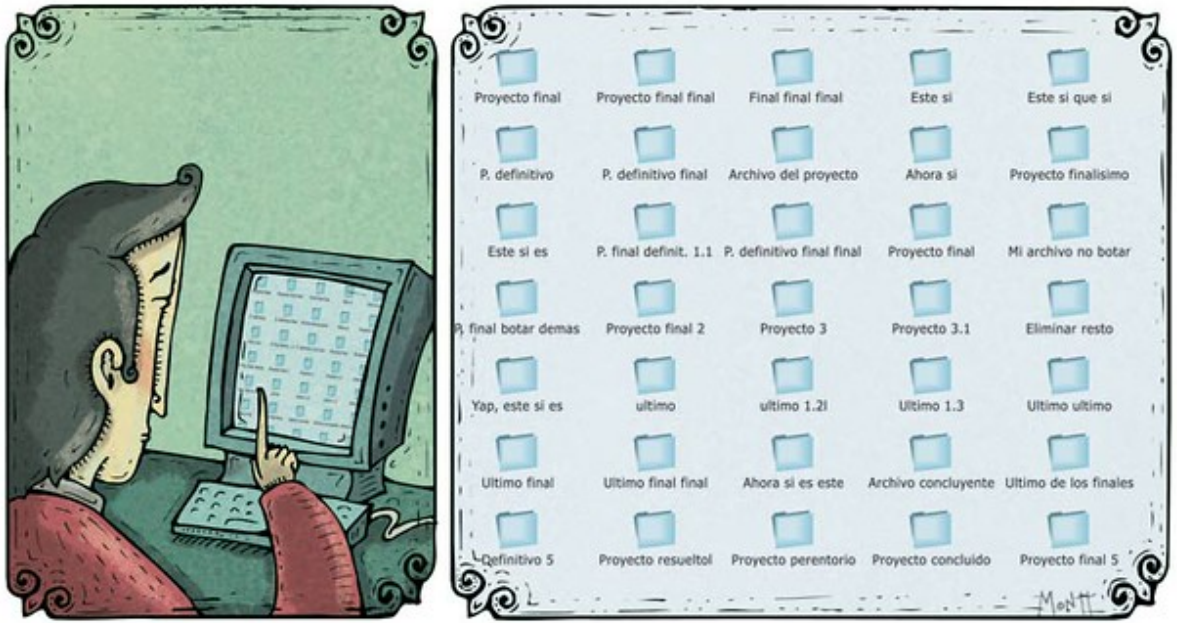
01

BLOQUE 1

Diagnóstico: lo que puede salir mal

El proyecto que todos hemos sufrido

Diagnóstico · introducción



Los 3 pecados capitales

Diagnóstico · 1 de 8

01 **Proyecto desordenado**

Archivos dispersos, sin convención ni estructura. Nadie sabe qué archivo es el bueno.

Los 3 pecados capitales

Diagnóstico · 1 de 8

01	**Proyecto desordenado**	Archivos dispersos, sin convención ni estructura. Nadie sabe qué archivo es el bueno.
02	**Malas prácticas de coding**	Un solo script de 500 líneas, rutas absolutas, sin comentarios, sin orden.

Los 3 pecados capitales

Diagnóstico · 1 de 8

01	**Proyecto desordenado**	Archivos dispersos, sin convención ni estructura. Nadie sabe qué archivo es el bueno.
02	**Malas prácticas de coding**	Un solo script de 500 líneas, rutas absolutas, sin comentarios, sin orden.
03	**Resultados producidos a mano**	Copiar y pegar números al Word, gráficos generados clic a clic.

Los 3 pecados capitales

Diagnóstico · 1 de 8

01	**Proyecto desordenado**	Archivos dispersos, sin convención ni estructura. Nadie sabe qué archivo es el bueno.
02	**Malas prácticas de coding**	Un solo script de 500 líneas, rutas absolutas, sin comentarios, sin orden.
03	**Resultados producidos a mano**	Copiar y pegar números al Word, gráficos generados clic a clic.

*A lo largo de la sesión vamos a **anclar cada herramienta a uno de estos problemas**.*

Pecado #1 — Proyecto desordenado

Diagnóstico · 2 de 8

- No sé cuál archivo contiene los datos originales.

Pecado #1 — Proyecto desordenado

Diagnóstico · 2 de 8

- No sé cuál archivo contiene los datos originales.
- Cambio algo y **no sé qué más deja de funcionar**.

Pecado #1 — Proyecto desordenado

Diagnóstico · 2 de 8

- No sé cuál archivo contiene los datos originales.
- Cambio algo y **no sé qué más deja de funcionar**.
- El jefe pregunta “¿puedes actualizar el reporte?” → toma 2 horas, 2 días o 2 semanas.

Pecado #1 — Proyecto desordenado

Diagnóstico · 2 de 8

- No sé cuál archivo contiene los datos originales.
- Cambio algo y **no sé qué más deja de funcionar**.
- El jefe pregunta “¿puedes actualizar el reporte?” → toma 2 horas, 2 días o 2 semanas.
- En 1 mes **ni yo sé** qué hice aquí.

Pecado #1 — Proyecto desordenado

Diagnóstico · 2 de 8

- No sé cuál archivo contiene los datos originales.
- Cambio algo y **no sé qué más deja de funcionar**.
- El jefe pregunta “¿puedes actualizar el reporte?” → toma 2 horas, 2 días o 2 semanas.
- En 1 mes **ni yo sé** qué hice aquí.
- Mi contraparte intenta reproducir mis resultados: **no puede**.

Pecado #1 — Proyecto desordenado

Diagnóstico · 2 de 8

- No sé cuál archivo contiene los datos originales.
- Cambio algo y **no sé qué más deja de funcionar**.
- El jefe pregunta “¿puedes actualizar el reporte?” → toma 2 horas, 2 días o 2 semanas.
- En 1 mes **ni yo sé** qué hice aquí.
- Mi contraparte intenta reproducir mis resultados: **no puede**.

Lo resolveremos con: *estructura de carpetas + GitHub + README.*

Pecado #2 — Malas prácticas de coding

Diagnóstico · 3 de 8 · el script de caos

```
library(tidyverse)
#library(dplyr)
datos <- read_csv("/Users/juan/Desktop/proyecto final/datos clientes USAR ESTE v3.csv")
datos2 = datos[!is.na(datos$nps),]
datos2$nps2 <- ifelse(datos2$nps > 50, "Promotor","otro")
metro <- datos2[datos2$region=="Metropolitana",]
mean(metro$nps); sd(metro$nps)
# valpo ← copiar-pegar 6 veces
Valpo <- datos2[datos2$region == "Valparaíso",]
mean(Valpo$nps); sd(Valpo$nps)
# OJO revisar esto
datos2$nps2 = ifelse(datos2$nps>=50,"promotor","Otro") # ← pisa lo de arriba!!
ggplot(datos2,aes(x=region,y=nps))+geom_boxplot()
lm(nps~satisfaccion,data=datos2)
```

Lo resolveremos con: *buenas prácticas de coding (Bloque 2).*

¿Cuántos problemas ves?

Diagnóstico · 4 de 8 · diagnóstico del script anterior

01 **Ruta absoluta con espacios** falla en cualquier otro computador.

¿Cuántos problemas ves?

Diagnóstico · 4 de 8 · diagnóstico del script anterior

01 **Ruta absoluta con espacios** falla en cualquier otro computador.

02 **Nombre “USAR ESTE v3”** control de versiones en el filename, no en código.

¿Cuántos problemas ves?

Diagnóstico · 4 de 8 · diagnóstico del script anterior

01	**Ruta absoluta con espacios**	falla en cualquier otro computador.
02	**Nombre “USAR ESTE v3”**	control de versiones en el filename, no en código.
03	**Sin separación**	limpieza + análisis + figura + modelo todo junto.

¿Cuántos problemas ves?

Diagnóstico · 4 de 8 · diagnóstico del script anterior

01	**Ruta absoluta con espacios**	falla en cualquier otro computador.
02	**Nombre “USAR ESTE v3”**	control de versiones en el filename, no en código.
03	**Sin separación**	limpieza + análisis + figura + modelo todo junto.
04	**Repetición manual**	`mean()`

¿Cuántos problemas ves?

Diagnóstico · 4 de 8 · diagnóstico del script anterior

01	**Ruta absoluta con espacios**	falla en cualquier otro computador.
02	**Nombre “USAR ESTE v3”**	control de versiones en el filename, no en código.
03	**Sin separación**	limpieza + análisis + figura + modelo todo junto.
04	**Repetición manual**	`mean()`
05	**La figura no se guarda**	desaparece cuando cierras R.

¿Cuántos problemas ves?

Diagnóstico · 4 de 8 · diagnóstico del script anterior

01	**Ruta absoluta con espacios**	falla en cualquier otro computador.
02	**Nombre “USAR ESTE v3”**	control de versiones en el filename, no en código.
03	**Sin separación**	limpieza + análisis + figura + modelo todo junto.
04	**Repetición manual**	`mean()`
05	**La figura no se guarda**	desaparece cuando cierras R.
06	**El modelo se imprime**	pero no se guarda ni se exporta.

¿Cuántos problemas ves?

Diagnóstico · 4 de 8 · diagnóstico del script anterior

01	**Ruta absoluta con espacios**	falla en cualquier otro computador.
02	**Nombre “USAR ESTE v3”**	control de versiones en el filename, no en código.
03	**Sin separación**	limpieza + análisis + figura + modelo todo junto.
04	**Repetición manual**	`mean()`
05	**La figura no se guarda**	desaparece cuando cierras R.
06	**El modelo se imprime**	pero no se guarda ni se exporta.
07	**Sin flags de progreso**	no sé si el script terminó o se colgó.

Pecado #3 — Resultados producidos a mano

Diagnóstico · 5 de 8 · dos problemas distintos, igual de graves

1) PROPENSO A ERRORES

Copiar un número mal, actualizar solo algunas celdas, una fórmula que referencia la columna equivocada. Errores invisibles que se acumulan.

Pecado #3 — Resultados producidos a mano

Diagnóstico · 5 de 8 · dos problemas distintos, igual de graves

1) PROPENSO A ERRORES

Copiar un número mal, actualizar solo algunas celdas, una fórmula que referencia la columna equivocada. Errores invisibles que se acumulan.

2) NO ESCALA

Si el análisis aplica a 6 regiones, tienes que repetir cada paso 6 veces. Si llegan datos nuevos el mes siguiente, todo desde cero.

Pecado #3 — Resultados producidos a mano

Diagnóstico · 5 de 8 · dos problemas distintos, igual de graves

1) PROPENSO A ERRORES

Copiar un número mal, actualizar solo algunas celdas, una fórmula que referencia la columna equivocada. Errores invisibles que se acumulan.

2) NO ESCALA

Si el análisis aplica a 6 regiones, tienes que repetir cada paso 6 veces. Si llegan datos nuevos el mes siguiente, todo desde cero.

*Ambos se resuelven con la misma idea: **outputs por código**.*

Propenso a errores — caso real

Diagnóstico · 6 de 8 · Reinhart & Rogoff (2010)

EL ERROR

Un error en una fórmula de Excel cambió los resultados de un paper que justificó **políticas de austeridad en media Europa**. El error existió por años sin que nadie lo detectara.

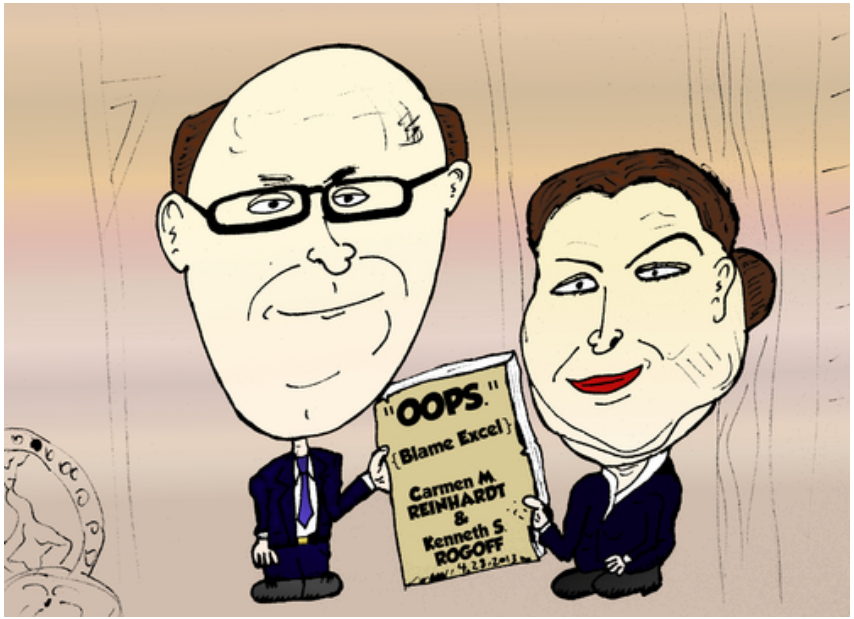
POR QUÉ IMPORTA

Si dos investigadores de Harvard pueden equivocarse en una celda de Excel y mover política pública, la pregunta no es *si* vas a cometer un error a mano — es *cuándo*.

La única defensa es que el resultado se pueda **regenerar desde cero con código auditable**. Si algo no calza, el error se encuentra.

Reinhart & Rogoff — el error que movió Europa

Diagnóstico · evidencia visual



Un error de fórmula en Excel → políticas de austeridad en media Europa.

No escalable — el caso típico

Diagnóstico · 7 de 8 · 6 regiones, reporte mensual

ENFOQUE MANUAL

1. Filtra la hoja por “Metropolitana”
2. Copia las tablas al Word
3. Actualiza los gráficos a mano
4. Repite para Valparaíso
5. Repite para Biobío
6. Repite para las 3 regiones restantes

→ 6 archivos Word distintos
→ Números copiados a mano
→ El mes siguiente: desde cero
→ **Garantía: habrá errores**

ENFOQUE CON CÓDIGO

```
regiones <- unique(
  clientes$region)

for (r in regiones) {
  quarto_render(
    "reporte.qmd",
    execute_params =
      list(region = r),
    output_file =
      paste0(r, ".html")
  )
}
```

→ *6 reportes idénticos en estructura*
→ *Mes siguiente: un clic*

Síntesis del diagnóstico

Diagnóstico · 8 de 8 · de qué nos vamos a hacer cargo

01 **Proyecto desordenado** → estructura de carpetas + GitHub + README.

Síntesis del diagnóstico

Diagnóstico · 8 de 8 · de qué nos vamos a hacer cargo

01 **Proyecto desordenado** → estructura de carpetas + GitHub + README.

02 **Malas prácticas de coding** → masterfile, scripts numerados, comentarios útiles, iteración.

Síntesis del diagnóstico

Diagnóstico · 8 de 8 · de qué nos vamos a hacer cargo

01	**Proyecto desordenado**	→ estructura de carpetas + GitHub + README.
02	**Malas prácticas de coding**	→ masterfile, scripts numerados, comentarios útiles, iteración.
03	**Resultados producidos a mano**	→ Quarto: texto + código + outputs en un solo archivo.

Síntesis del diagnóstico

Diagnóstico · 8 de 8 · de qué nos vamos a hacer cargo

01	**Proyecto desordenado**	→ estructura de carpetas + GitHub + README.
02	**Malas prácticas de coding**	→ masterfile, scripts numerados, comentarios útiles, iteración.
03	**Resultados producidos a mano**	→ Quarto: texto + código + outputs en un solo archivo.

*Cada herramienta del Bloque 2 **ataca un pecado específico**. No es magia, es ingeniería.*

02

BLOQUE 2

Principios y herramientas

Tres principios que no cambian, y un set de herramientas que los hacen posibles. Cada herramienta resuelve ***un problema concreto****.*

¿Qué es el *workflow*?

DEFINICIÓN

Un **workflow** es el flujo de producción de un análisis: el camino que va desde los datos crudos hasta el reporte final.

¿Qué es el *workflow*?

Principios · 1 de 4

DEFINICIÓN

Un **workflow** es el flujo de producción de un análisis: el camino que va desde los datos crudos hasta el reporte final.

EL OBJETIVO

Si mi workflow es efectivo, puedo replicar todos mis resultados —figuras, tablas, informes— con un solo clic o una línea de código.

¿Qué es el *workflow*?

Principios · 1 de 4

DEFINICIÓN

Un **workflow** es el flujo de producción de un análisis: el camino que va desde los datos crudos hasta el reporte final.

EL OBJETIVO

Si mi workflow es efectivo, puedo replicar todos mis resultados —figuras, tablas, informes— con un solo clic o una línea de código.

Esto requiere tres cosas: **estructura predecible**, **datos crudos intocables**, **outputs siempre por código**.

Principio #1 — Estructura predecible

Principios · 2 de 4

PROBLEMA QUE RESUELVE

“¿cuál archivo era el bueno?” · “¿dónde está el dato?” · “¿qué hace este script?”

```
estudio_satisfaccion_2025/
|
+- data/
|  +- raw/        # datos del cliente · NUNCA se modifican
|  +- clean/      # output de los scripts de limpieza
|
+- code/
|  +- 01_clean.R    # pipeline de limpieza
|  +- 02_analysis.R  # análisis y modelos
|  +- 03_figures.R   # generación de gráficos
|
+- output/
|  +- figures/     # .png, .pdf
|  +- tables/      # .csv para reportes
|
+- docs/reporte.qmd   # documento maestro
+- README.md          # documentación del proyecto
```

Principio #2 — Raw vs. Clean

Principios · 3 de 4

PROBLEMA QUE RESUELVE

“¿qué le hicieron a estos datos?” · “¿por qué la suma cambió?”

✓ EL FLUJO CORRECTO

```
# 01_clean.R
raw <- read_csv("data/raw/clientes.csv")

clean <- raw |>
  filter(!is.na(nps)) |>
  mutate(
    nps_grupo = case_when(
      nps < 0  ~ "Detractor",
      nps < 50 ~ "Pasivo",
      TRUE     ~ "Promotor"
    )
  )

write_csv(clean,
  "data/clean/clientes.csv")
```

× LO QUE NUNCA HACER

1. Abrir el Excel del cliente
2. Borrar las filas “raras”
3. Cambiar formatos a mano
4. Guardar como `clientes_USAR_ESTE.xlsx`

→ las transformaciones quedan invisibles, irreproducibles e imposibles de auditar.

Principio #3 — Outputs por código

Principios · 4 de 4

PROBLEMA QUE RESUELVE

“actualiza el reporte con los nuevos datos” → 2 horas de copiar-pegar.

Cada gráfico, tabla, número que aparece en el reporte final debe poder **regenerarse corriendo un script**.

```r
# Corre el pipeline completo
source("0_masterfile.R")

# Genera la figura
ggsave("output/figures/nps_region.png", plot = p, width = 8, height = 5)

# Genera la tabla
write_csv(resumen_regional, "output/tables/resumen_regional.csv")

# Genera el reporte completo
quarto::quarto_render("docs/reporte.qmd")
```

Resumen — problema → herramienta

Herramientas · mapa

Problema	Herramienta / práctica
Proyecto desordenado	**Estructura de carpetas predecible**
No sé qué cambió ni cuándo	**GitHub**
Nadie entiende mi proyecto	**README**
Datos sucios	**Tidy data + script** `01_clean.R`
Repetición manual	**Iteración (**`for` **loops)**
“Actualizá el reporte” toma 2 horas	**Quarto + parámetros**
Necesito ayuda con código	**IA integrada al pipeline**

Práctica #1 — El masterfile

Buenas prácticas · 1 de 8

PROBLEMA QUE RESUELVE

"¿en qué orden corro los scripts?" · "¿cuál era el punto de entrada?"

Un **masterfile** (`1_masterfile.R`) es el único archivo que se corre directamente. Llama a todos los demás **en orden**.

```
# 1. Limpiar el entorno
cat("\014"); rm(list = ls())

# 2. Cargar packages
library(pacman); p_load(tidyverse, knitr, stargazer)

# 3. Definir directorios (solo aquí, una vez)
folder    <- "/Users/bucca/proyecto_nps/"
dircode   <- paste0(folder, "code/")
dirdata   <- paste0(folder, "data/")
dirresults <- paste0(folder, "results/")

# 4. Correr el pipeline en orden
source(paste0(dircode, "2_exploration.R"))
source(paste0(dircode, "3_recoding.R"))
source(paste0(dircode, "4_analyses.R"))
```

Práctica #2 — Scripts numerados, responsabilidad única

Buenas prácticas · 2 de 8

PROBLEMA QUE RESUELVE

"¿dónde estaba la limpieza?" · "¿qué hace este script?"

```
code/
+- 1_masterfile.R      # llama a todos los demás
+- 2_exploration.R     # solo exploración
+- 3_recoding.R        # solo recodificación
+- 4_analyses.R        # análisis y outputs globales
+- 5_analyses_bysucursal.R  # análisis por subgrupo
```

- Cada script hace **una sola cosa**. Los números marcan el **orden de ejecución**.
- Si algo falla en el análisis, sabes que el problema está en `4_analyses.R`.
- Si quieres cambiar una recodificación, vas directo a `3_recoding.R`.

Práctica #3 — Directorios al inicio, rutas relativas

Buenas prácticas · 3 de 8

PROBLEMA QUE RESUELVE

“en mi computador funciona pero en el tuyo no.”

× RUTA ABSOLUTA DISPERSA

```
# en 2_exploration.R
datos <- read_dta(
  "/Users/juan/Desktop/
   proyecto final/datos v3.dta"
)

# en 4_analyses.R
ggsave(
  "/Users/juan/Desktop/
   proyecto final/fig1_FINAL.pdf"
)
```

✓ DIRECTORIOS UNA SOLA VEZ

```
# en 1_masterfile.R
folder    <- "/Users/bucca/nps/"
dirdata   <- paste0(folder, "data/")
dirresults <- paste0(folder, "results/")

# en 4_analyses.R
datos <- read_dta(
  paste0(dirdata, "clientes.dta"))
ggsave(
  paste0(dirresults, "figura1.pdf"))
```

Para correr el proyecto en otro computador: ***cambiar una sola línea****.*

Práctica #4 — Guardar todos los outputs

Buenas prácticas · 4 de 8

PROBLEMA QUE RESUELVE

“calculé el modelo pero no sé dónde quedó” · “la figura solo existe en el panel de R”

```
# - Figuras -------------------
mi_figura <- clientes |>
  ggplot(aes(x = region, y = nps, fill = region)) +
  geom_boxplot(alpha = 0.7) + theme(legend.position = "none")
ggsave(paste0(dirresults, "figura_nps_region.pdf"),
      mi_figura, width = 15, height = 10, units = "cm")
# - Tablas --------------------
stargazer(model1, model2, type = "text",
        out = paste0(dirresults, "tabla_regresion.txt"))
# - Modelos -------------------
saveRDS(model1, paste0(dirresults, "modelo_nps.rds"))
```

*Si no está guardado en `results/`, **no existe**.*

Práctica #5 — Flags de debugging

Buenas prácticas · 5 de 8

PROBLEMA QUE RESUELVE

“el masterfile corrió 10 minutos y no sé si terminó o se colgó.”

Al final de cada script, una línea que confirme que llegó hasta ahí.

```
# al final de 2_exploration.R
cat("======== EXPLORACIÓN LISTA ========\n")

# al final de 3_recoding.R
cat("======== RECODIFICACIÓN LISTA ========\n")

# al final de 4_analyses.R
cat("======== ANÁLISIS LISTOS ========\n")
```

Práctica #6 — Comentar el código (el *por qué*)

Buenas prácticas · 6 de 8

Un buen comentario no explica **qué** hace el código (eso ya se ve); explica **por qué** se tomó esa decisión.

× COMENTARIO INÚTIL

```
# filtra filas con NA
datos <- datos |> filter(!is.na(nps))

# calcula media
media <- mean(datos$nps)

# hace gráfico
ggplot(datos, aes(x = nps)) +
  geom_histogram()
```

✓ COMENTARIO ÚTIL

```
# NAs en nps son MCAR según
# exploración (2_exploration.R l.34).
# Son 7% del total -- aceptable.
datos <- datos |> filter(!is.na(nps))

# Media usada como benchmark en el
# reporte ejecutivo (reporte.qmd §2)
media <- mean(datos$nps)
```

Práctica #7 — Estructura del script

Buenas prácticas · 7 de 8 · encabezados de sección

```
# ========================================
# 4_analyses.R
# Análisis NPS por región y sector
# Autor: Mauricio Bucca · mayo 2026
# Inputs:  data/clean/clientes.csv
# Outputs: results/figura_nps.pdf, results/tabla_regresion.txt
# ========================================

# - 1. Cargar datos ------------
clientes <- read_csv(paste0(dirdata, "clientes.csv"))

# - 2. Descriptivos ------------
# NPS por región -- tabla de resumen
clientes |> group_by(region) |>
  summarise(media = mean(nps, na.rm = TRUE)) |> arrange(desc(media))

# - 3. Modelo ---------------
# OLS simple (no mixto): n < 30 por grupo, sin estructura jerárquica suficiente.
model1 <- lm(nps ~ satisfaccion + facturacion_mm, data = clientes)
```

Práctica #8 — Iterar en vez de copiar-pegar

Buenas prácticas · 8 de 8

PROBLEMA QUE RESUELVE

“el análisis hay que repetirlo para 13 sucursales.”

En vez de duplicar `4_analyses.R` trece veces, el masterfile itera.

```r
# 1_masterfile.R
sucursales <- unique(clientes$sucursal)  # c("Santiago", "Valpo", ...)

for (s in sucursales) {
  cat("=== ANALIZANDO:", s, "===\n")
  clientes_suc <- clientes |> filter(sucursal == s)
  source(paste0(dircode, "5_analyses_bysucursal.R"))
}
```

Si se agrega una sucursal, el código la maneja solo.

El antes y el después

Buenas prácticas · síntesis

× MALAS PRÁCTICAS

```
analisis.R      # 800 líneas
analisis_v2.R
analisis_FINAL.R

Ruta: /Users/juan/Desktop/
  proyecto final/datos v3.csv

mean(metro$nps)  # repetido 5x
mean(valpo$nps)
mean(biobio$nps)
...

# figura → solo en el panel,
# perdida al cerrar R
```

✓ BUENAS PRÁCTICAS

```
1_masterfile.R   # entrada
2_exploration.R
3_recoding.R
4_analyses.R
5_analyses_by*.R

folder <- "miruta/"  # una vez

for (s in sucursales) {  # itera
  source("5_analyses_bysuc.R")
}

ggsave(paste0(dirresults,
       "figura.pdf"))  # guardado
```

El ecosistema moderno

Herramientas · ¿qué problema resuelve cada una?

Herramienta	¿Qué problema resuelve?
R	Python

El ecosistema moderno

Herramientas · ¿qué problema resuelve cada una?

Herramienta	¿Qué problema resuelve?
R	Python
Positron (IDE)	Una sola ventana para todo el workflow.

El ecosistema moderno

Herramientas · ¿qué problema resuelve cada una?

Herramienta	¿Qué problema resuelve?
R	Python
Positron (IDE)	Una sola ventana para todo el workflow.
GitHub	“¿qué cambió?”, “¿quién lo cambió?”, “volvamos a la versión del lunes”.

El ecosistema moderno

Herramientas · ¿qué problema resuelve cada una?

Herramienta	¿Qué problema resuelve?
R	Python
Positron (IDE)	Una sola ventana para todo el workflow.
GitHub	“¿qué cambió?”, “¿quién lo cambió?”, “volvamos a la versión del lunes”.
Quarto	Reportes y presentaciones reproducibles que se regeneran con un comando.

El ecosistema moderno

Herramientas · ¿qué problema resuelve cada una?

Herramienta	¿Qué problema resuelve?
R	Python
Positron (IDE)	Una sola ventana para todo el workflow.
GitHub	“¿qué cambió?”, “¿quién lo cambió?”, “volvamos a la versión del lunes”.
Quarto	Reportes y presentaciones reproducibles que se regeneran con un comando.
IA (Claude Code, etc.)	Asistencia dentro del pipeline (no fuera).

R o Python — herramientas, no religión

Herramientas · 1

Los **principios** del workflow no cambian con el lenguaje.

Tarea	R	Python
Estadística clásica · ggplot · reportes	✓✓✓	✓
Manipulación tabular	✓✓✓	✓✓
Machine Learning · NLP · scraping · automatización	✓	✓✓✓

*Para estudios de mercado: **R con tidyverse** suele ser el camino más corto.*

Positron — el IDE moderno

Herramientas · 2 · una ventana para todo el workflow

PROBLEMA QUE RESUELVE

“tengo R abierto en RStudio, Python en VS Code, Quarto en otra ventana, y la terminal aparte.”

LENGUAJES

- Consolas **R y Python simultáneas**
- Cambio fluido entre uno y otro
- Variable explorer para ambos
- Data frame viewer integrado

WORKFLOW

- **Quarto** nativo
- **GitHub** integrado
- **Terminal** integrada
- **Claude Code** y otros asistentes IA
- Extensiones de VS Code

Todo el workflow en una sola ventana.

*En Positron todo esto se hace **con botones**, no con terminal.*

GitHub — la máquina del tiempo del proyecto

Herramientas · 3a

PROBLEMA QUE RESUELVE

“¿qué cambió desde ayer?” · “rompiste algo y no sé qué” · “tengo 5 versiones del mismo archivo.”

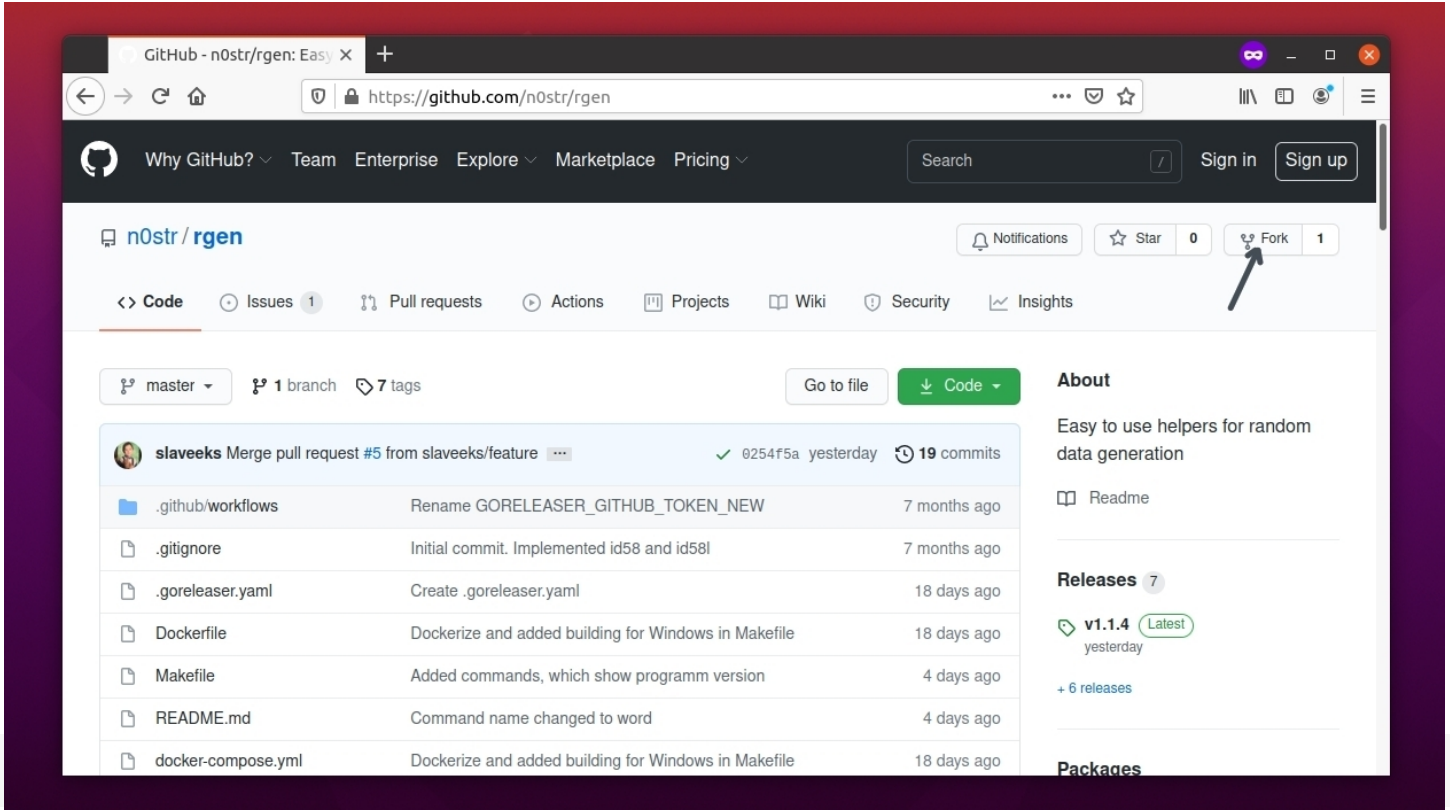
QUÉ ES

Una plataforma para guardar y versionar proyectos en la nube. Guarda **todo el historial**, no solo la versión actual. Tú decides cuándo tomar una “foto” y le pones un mensaje descriptivo.

01	**Repositorio**	Como una carpeta nueva, pero en la nube.
02	**Conectar a Positron**	Un clic y queda enlazado a tu computador.
03	**Commit**	Una foto del proyecto con un mensaje tuyo.
04	**Pull**	Push

¿Qué se ve en GitHub?

Herramientas · 3 · una interfaz web encima del repositorio



El README — tu yo del futuro lo agradecerá

Herramientas · 4a

PROBLEMA QUE RESUELVE

“abro el proyecto 1 semana después y no entiendo qué hice.”

¿QUÉ RESPONDE?

- ¿Qué hace este proyecto?
- ¿Cómo lo corro desde cero?
- ¿Dónde están los datos?
- ¿Qué dependencias necesito?

¿PARA QUIÉN?

- **Tú en 1 semana**
- El colega que toma el proyecto
- El cliente que quiere replicar
- La IA que te ayudará

README — cómo replicar (el corazón)

Herramientas · 4b · los pasos deben funcionar en cualquier computador

```
## Cómo replicar desde cero

### 1. Clonar el repositorio
git clone https://github.com/dobleaLAB/diagnostico_nps.git
cd diagnostico_nps

### 2. Instalar dependencias
renv::restore()  # instala los mismos packages y versiones

### 3. Correr el pipeline completo
source("code/01_clean.R")    # limpia y estandariza datos
source("code/02_analysis.R") # análisis exploratorio y modelos
source("code/03_figures.R")  # genera gráficos en output/figures/
quarto render docs/reporte.qmd # produce el reporte final

### 4. Reportes regionales (opcional)
source("code/04_render_regiones.R") # 6 reportes, uno por región
```

Iteración — principio DRY

Herramientas · 5 · Don’t Repeat Yourself

LA REGLA

Si copias y pegas código más de **dos veces**, hay una mejor manera. Menos código → menos errores → más confianza.

× REPETICIÓN MANUAL

```
media_metro  <- mean(
  clientes$nps[clientes$region ==
    "Metropolitana"], na.rm=T)
media_valpo  <- mean(
  clientes$nps[clientes$region ==
    "Valparaíso"], na.rm=T)
media_biobio <- mean(
  clientes$nps[clientes$region ==
    "Biobío"], na.rm=T)
# ¿y si son 6, 16, 60?
```

Iteración — principio DRY

Herramientas · 5 · Don't Repeat Yourself

LA REGLA

Si copias y pegas código más de **dos veces**, hay una mejor manera. Menos código → menos errores → más confianza.

× REPETICIÓN MANUAL

```
media_metro  <- mean(
  clientes$nps[clientes$region ==
    "Metropolitana"], na.rm=T)
media_valpo  <- mean(
  clientes$nps[clientes$region ==
    "Valparaíso"], na.rm=T)
media_biobio <- mean(
  clientes$nps[clientes$region ==
    "Biobío"], na.rm=T)
# ¿y si son 6, 16, 60?
```

✓ FOR LOOP

```
for (r in unique(
    clientes$region)) {
  media <- mean(
    clientes$nps[
      clientes$region == r],
    na.rm = TRUE)
  cat(r, ":",
      round(media, 1), "\n")
}
# 6, 16 o 60: igual.
```

Quarto — ¿qué problema resuelve?

Herramientas · 6a · reportes y presentaciones reproducibles

“Hay que actualizar el reporte con los datos de mayo” → 2 horas de copiar-pegar.

“El cliente quiere un reporte por cada una de sus 6 regiones” → ¿hago 6 Word distintos?

“Necesito que el gráfico y la tabla cuadren con el texto” → siempre se desincronizan.

LA IDEA

Quarto une **texto + código + resultados** en un solo archivo. Cambian los datos → todo el reporte se regenera solo.

Quarto — texto + código + outputs

Herramientas · 6b · un mismo `.qmd`, muchos formatos de salida

Texto

Markdown, ecuaciones, imágenes, tablas

Código

Chunks de R, Python, Julia, Bash — o los cuatro

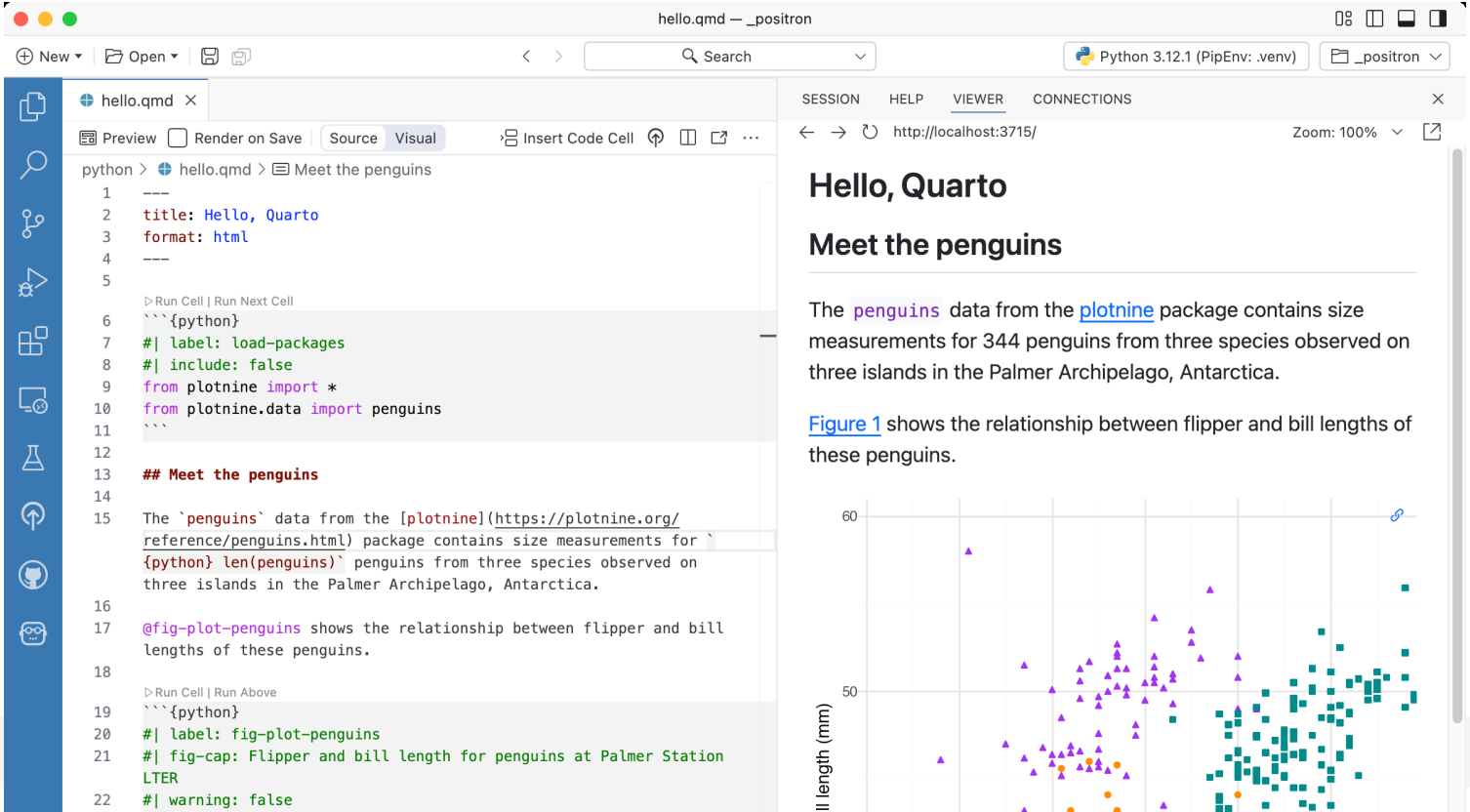
Output

HTML, PDF, Word, PPTX, slides, dashboards

Output	Para qué sirve
HTML	Reporte interactivo para mail
PDF	Documento final formal · papers · entregables.
Word (.docx)	Cuando el cliente edita en Word y no se mueve.
Reveal.js	PPTX

Quarto en Positron — un solo archivo

Herramientas · 5 · código a la izquierda, preview a la derecha



Anatomía de un documento Quarto

Herramientas · 6c · YAML + texto + chunks

```
---
title:  "Estudio NPS · Metropolitana"
author: "DobleA LAB"
format: html          # acá decides el output
---

## Resumen

La muestra contiene 150 empresas en 6 regiones.

"‘{r}
library(tidyverse)
clientes <- read_csv("data/clean/clientes.csv")
mean(clientes$nps, na.rm = TRUE)
"‘

El gráfico siguiente muestra la distribución por región...
```

*Tres bloques: **YAML** (configuración), **texto markdown**, **chunks de código**.*

Valores inline — el número vive en el código

Herramientas · 6d · nunca más copiar resultados a mano

```
La muestra contiene `r nrow(clientes)` empresas
de `r n_distinct(clientes$region)` regiones.

El NPS promedio es `r round(mean(clientes$nps, na.rm=TRUE), 1)`.
```

IMPLICANCIA

Cambian los datos → **los números del texto se actualizan solos**. El número y la fuente de verdad nunca se desincronizan.

La feature que cambia todo — parámetros

Herramientas · 6e · un mismo documento, muchas variantes

PROBLEMA

6 regiones × un reporte por cada una = 6 reportes idénticos en estructura, distintos en datos.

DECLARAR (en YAML)

```
---
title: "Estudio Regional"
params:
  region: "Metropolitana"
  año: 2026
  umbral_promotor: 50
---
```

USAR (en el cuerpo)

```
## Región: `r params$region`

"`{r}
datos <- clientes |>
  filter(region == params$region,
         year == params$año)
"`

NPS: **`r round(mean(
   datos$nps, na.rm=T),1)`**
```

Generar todos los reportes con un loop

Herramientas · 6f · Quarto + iteración

```
library(quarto)

regiones <- unique(clientes$region)

for (r in regiones) {
  quarto_render(
    input       = "docs/reporte_regional.qmd",
    execute_params = list(region = r, año = 2026),
    output_file   = paste0("reporte_", tolower(r), ".html")
  )
}
```

OUTPUT

```
✓ reporte_metropolitana.html    ✓ reporte_valparaíso.html
✓ reporte_biobío.html          ✓ reporte_antofagasta.html
✓ reporte_araucanía.html        ✓ reporte_los_lagos.html
```

IA en el workflow

Herramientas · 7 · asistencia dentro del pipeline, no fuera

PROBLEMA QUE RESUELVE

“me quedé pegado con un bug y voy a perder 2 horas googleando.”

ASISTENTE EN EL IDE

- Chat dentro de Positron / VS Code
- Accede al contexto del archivo abierto
- Ideal para **debugging interactivo**
- Sugerencias mientras escribes

IA en el workflow

Herramientas · 7 · asistencia dentro del pipeline, no fuera

PROBLEMA QUE RESUELVE

“me quedé pegado con un bug y voy a perder 2 horas googleando.”

ASISTENTE EN EL IDE

- Chat dentro de Positron / VS Code
- Accede al contexto del archivo abierto
- Ideal para **debugging interactivo**
- Sugerencias mientras escribes

CLAUDE CODE (TERMINAL)

- Corre desde la terminal del IDE
- Lee y escribe archivos
- Puede ejecutar código
- Ideal para **tareas largas**

IA en el workflow

Herramientas · 7 · asistencia dentro del pipeline, no fuera

PROBLEMA QUE RESUELVE

“me quedé pegado con un bug y voy a perder 2 horas googleando.”

ASISTENTE EN EL IDE

- Chat dentro de Positron / VS Code
- Accede al contexto del archivo abierto
- Ideal para **debugging interactivo**
- Sugerencias mientras escribes

CLAUDE CODE (TERMINAL)

- Corre desde la terminal del IDE
- Lee y escribe archivos
- Puede ejecutar código
- Ideal para **tareas largas**

*La IA **opera dentro** del pipeline. No lo define. No lo reemplaza.*

03

BLOQUE 3

Todo junto: un ejemplo aplicado

Recapitulación con un caso concreto, paso a paso, mostrando qué herramienta resuelve qué problema en el camino.

La situación

Ejemplo · 1 de 8 · lunes a las 9 AM

EL ENCARGO

El cliente envía la encuesta NPS de mayo de **150 empresas en 6 regiones**.

- Un **reporte general** con el NPS global.

Veamos cómo se ve este trabajo con un buen workflow.

La situación

Ejemplo · 1 de 8 · lunes a las 9 AM

EL ENCARGO

El cliente envía la encuesta NPS de mayo de **150 empresas en 6 regiones**.

- Un **reporte general** con el NPS global.
- Un **reporte por región** (6 documentos).

Veamos cómo se ve este trabajo con un buen workflow.

La situación

Ejemplo · 1 de 8 · lunes a las 9 AM

EL ENCARGO

El cliente envía la encuesta NPS de mayo de **150 empresas en 6 regiones**.

- Un **reporte general** con el NPS global.
- Un **reporte por región** (6 documentos).
- Todo en **HTML** para subir a su intranet.

Veamos cómo se ve este trabajo con un buen workflow.

La situación

Ejemplo · 1 de 8 · lunes a las 9 AM

EL ENCARGO

El cliente envía la encuesta NPS de mayo de **150 empresas en 6 regiones**.

- Un **reporte general** con el NPS global.
- Un **reporte por región** (6 documentos).
- Todo en **HTML** para subir a su intranet.
- Para el **viernes**.

Veamos cómo se ve este trabajo con un buen workflow.

Paso 0 — La estructura ya existe

Ejemplo · 2 de 8 · no empieza desde cero

```
estudio_nps_2026/
+- data/
|   +- raw/           # acá pega la encuesta de mayo
|   +- clean/
+- code/
|   +- 01_clean.R
|   +- 02_analysis.R
|   +- 03_render_reportes.R
+- docs/
|   +- reporte_general.qmd
|   +- reporte_regional.qmd   # parametrizado por región
+- output/
+- README.md
```

Ya está armado de proyectos anteriores. ***Mismo patrón en cada proyecto****.*

Paso 1 — Datos crudos al lugar correcto

Ejemplo · 3 de 8

01 Copio el archivo del cliente a `data/raw/`.

Paso 1 — Datos crudos al lugar correcto

Ejemplo · 3 de 8

01 Copio el archivo del cliente a `data/raw/`.

02 Desde Positron, hago un **commit** con el mensaje: *“data: agregar encuesta mayo 2026”*.

Paso 1 — Datos crudos al lugar correcto

Ejemplo · 3 de 8

01 Copio el archivo del cliente a `data/raw/`.

02 Desde Positron, hago un **commit** con el mensaje: *“data: agregar encuesta mayo 2026”*.

03 Hago **push** para subirlo a GitHub.

Paso 1 — Datos crudos al lugar correcto

Ejemplo · 3 de 8

01 Copio el archivo del cliente a `data/raw/`.

02 Desde Positron, hago un **commit** con el mensaje: *“data: agregar encuesta mayo 2026”*.

03 Hago **push** para subirlo a GitHub.

HERRAMIENTAS USADAS

Estructura de carpetas (`raw/`) + GitHub (respaldo y trazabilidad). El archivo queda intocado, nunca se edita a mano.

Paso 2 — Limpieza vía script

Ejemplo · 4 de 8

```r
# code/01_clean.R
library(tidyverse)
raw <- read_csv("data/raw/encuesta_mayo_2026.csv")
clean <- raw |>
  filter(!is.na(nps)) |>
  mutate(
    nps_grupo = case_when(
      nps < 0  ~ "Detractor",
      nps < 50 ~ "Pasivo",
      TRUE     ~ "Promotor"
    )
  )
write_csv(clean, "data/clean/clientes.csv")
```

HERRAMIENTAS

tidyverse, tidy data, principio raw → clean.

Paso 3 — Reporte regional parametrizado

Ejemplo · 5 de 8

```
---
title: "Estudio NPS · `r params$region`"
format: html
params:
  region: "Metropolitana"
---

"`{r}
library(tidyverse)
clientes <- read_csv("data/clean/clientes.csv")
datos <- clientes |> filter(region == params$region)
"`

## Resumen

En **`r params$region`** medimos `r nrow(datos)` empresas.
El NPS promedio es **`r round(mean(datos$nps, na.rm=TRUE), 1)`**.
```

Herramientas: *Quarto, parámetros, valores inline, chunks.*

Paso 4 — Generar los 6 reportes con un loop

Ejemplo · 6 de 8

```r
# code/03_render_reportes.R
library(quarto)

clientes <- read_csv("data/clean/clientes.csv")

for (r in unique(clientes$region)) {
  quarto_render(
    input       = "docs/reporte_regional.qmd",
    execute_params = list(region = r),
    output_file   = paste0("reporte_", tolower(r), ".html")
  )
}
```

RESULTADO

~**40 segundos después**: 6 reportes listos. Cero copiar-pegar.
Herramientas: iteración (`for`), Quarto con parámetros.

Paso 5 — Commit y entrega

Ejemplo · 7 de 8

01 Desde Positron, hago **commit** con el mensaje: *“feat: reportes NPS mayo 2026”*.

Paso 5 — Commit y entrega

Ejemplo · 7 de 8

01 Desde Positron, hago **commit** con el mensaje: *“feat: reportes NPS mayo 2026”*.

02 Hago **push** para subir todo a GitHub.

Paso 5 — Commit y entrega

Ejemplo · 7 de 8

01 Desde Positron, hago **commit** con el mensaje: *“feat: reportes NPS mayo 2026”*.

02 Hago **push** para subir todo a GitHub.

03 Comparto el link de los reportes con el cliente.

Paso 5 — Commit y entrega

Ejemplo · 7 de 8

01 Desde Positron, hago **commit** con el mensaje: *“feat: reportes NPS mayo 2026”*.

02 Hago **push** para subir todo a GitHub.

03 Comparto el link de los reportes con el cliente.

BONUS

Si el cliente pide cambios el jueves, ***todo el pipeline corre de nuevo en ~1 minuto****.*

¿Qué pasó en el camino?

Ejemplo · 8 de 8 · comparación de tiempos

Paso	Manual	Workflow
Limpiar datos	~1 hora	5 seg (script)

¿Qué pasó en el camino?

Ejemplo · 8 de 8 · comparación de tiempos

Paso	Manual	Workflow
Limpiar datos	~1 hora	5 seg (script)
Generar 6 reportes	~3 horas	40 seg (loop)

¿Qué pasó en el camino?

Ejemplo · 8 de 8 · comparación de tiempos

Paso	Manual	Workflow
Limpiar datos	~1 hora	5 seg (script)
Generar 6 reportes	~3 horas	40 seg (loop)
Detectar y arreglar un error	~2 horas	1 min (re-render)

¿Qué pasó en el camino?

Ejemplo · 8 de 8 · comparación de tiempos

Paso	Manual	Workflow
Limpiar datos	~1 hora	5 seg (script)
Generar 6 reportes	~3 horas	40 seg (loop)
Detectar y arreglar un error	~2 horas	1 min (re-render)
Documentar qué se hizo	~30 min	ya está en el código

¿Qué pasó en el camino?

Ejemplo · 8 de 8 · comparación de tiempos

Paso	Manual	Workflow
Limpiar datos	~1 hora	5 seg (script)
Generar 6 reportes	~3 horas	40 seg (loop)
Detectar y arreglar un error	~2 horas	1 min (re-render)
Documentar qué se hizo	~30 min	ya está en el código
TOTAL	**~7 horas**	**~5 min**

¿Qué pasó en el camino?

Ejemplo · 8 de 8 · comparación de tiempos

Paso	Manual	Workflow
Limpiar datos	~1 hora	5 seg (script)
Generar 6 reportes	~3 horas	40 seg (loop)
Detectar y arreglar un error	~2 horas	1 min (re-render)
Documentar qué se hizo	~30 min	ya está en el código
TOTAL	**~7 horas**	**~5 min**

*El cliente recibe lo mismo. Tú hiciste **80 veces menos trabajo**.*

Y cuando llegue la encuesta de junio: mismo flujo, mismo tiempo.

Los 5 hábitos que importan

Cierre · lo que se queda contigo

01 **Estructura ANTES de código** — El mismo patrón en cada proyecto.

Los 5 hábitos que importan

Cierre · lo que se queda contigo

01	**Estructura ANTES de código**	El mismo patrón en cada proyecto.
02	**Datos crudos intocables**	Si cambia algo, lo hace el código.

Los 5 hábitos que importan

Cierre · lo que se queda contigo

01	**Estructura ANTES de código**	El mismo patrón en cada proyecto.
02	**Datos crudos intocables**	Si cambia algo, lo hace el código.
03	**Outputs siempre por código**	Nunca a mano.

Los 5 hábitos que importan

Cierre · lo que se queda contigo

01	**Estructura ANTES de código**	El mismo patrón en cada proyecto.
02	**Datos crudos intocables**	Si cambia algo, lo hace el código.
03	**Outputs siempre por código**	Nunca a mano.
04	**Commits frecuentes**	Con mensajes descriptivos.

Los 5 hábitos que importan

Cierre · lo que se queda contigo

01	**Estructura ANTES de código**	El mismo patrón en cada proyecto.
02	**Datos crudos intocables**	Si cambia algo, lo hace el código.
03	**Outputs siempre por código**	Nunca a mano.
04	**Commits frecuentes**	Con mensajes descriptivos.
05	**Documentación como parte del flujo**	No al final.

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?
- [ ] ¿Un colega puede clonarlo y correrlo?

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?
- [ ] ¿Un colega puede clonarlo y correrlo?
- [ ] ¿Está documentado de dónde vienen los datos?

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?
- [ ] ¿Un colega puede clonarlo y correrlo?
- [ ] ¿Está documentado de dónde vienen los datos?
- [ ] ¿Está claro qué decisiones tomé (filtros, transformaciones)?

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?
- [ ] ¿Un colega puede clonarlo y correrlo?
- [ ] ¿Está documentado de dónde vienen los datos?
- [ ] ¿Está claro qué decisiones tomé (filtros, transformaciones)?
- [ ] ¿Los gráficos y tablas se generan con código?

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?
- [ ] ¿Un colega puede clonarlo y correrlo?
- [ ] ¿Está documentado de dónde vienen los datos?
- [ ] ¿Está claro qué decisiones tomé (filtros, transformaciones)?
- [ ] ¿Los gráficos y tablas se generan con código?
- [ ] ¿Hay un README?

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?
- [ ] ¿Un colega puede clonarlo y correrlo?
- [ ] ¿Está documentado de dónde vienen los datos?
- [ ] ¿Está claro qué decisiones tomé (filtros, transformaciones)?
- [ ] ¿Los gráficos y tablas se generan con código?
- [ ] ¿Hay un README?
- [ ] ¿El proyecto está versionado en GitHub?

Checklist antes de enviar un análisis

Cierre · 8 preguntas para tu yo del futuro

- [ ] ¿Puedo reproducir TODO con un solo comando?
- [ ] ¿Un colega puede clonarlo y correrlo?
- [ ] ¿Está documentado de dónde vienen los datos?
- [ ] ¿Está claro qué decisiones tomé (filtros, transformaciones)?
- [ ] ¿Los gráficos y tablas se generan con código?
- [ ] ¿Hay un README?
- [ ] ¿El proyecto está versionado en GitHub?
- [ ] ¿Si cambian los datos, puedo señalar exactamente dónde cambió cada número?

CLASE 3 — CIERRE

Gracias

MAURICIO BUCCA

Sociología UC · DobleA LAB

mebucca.github.io

github.com/mebucca

mebucca@gmail.com

DOBLEA LAB

Lunes 19 de mayo, 2026

Clase 3 · Taller de Investigación Aplicada