DevRep

April 24, 2024

[ ]:
```
---
title: "Replication: Urban Public Works in Spatial Equilibrium"
author: "Bo Jacobs Strom"
date: "April 2024"
format:
    html:
        code-overflow: wrap
        embed-resources: true
execute:
    output: false
    echo: false
---
```

This quarto document contains my replication of the main results of *Urban Public Works in Spatial Equilibrium (2023)* by *Franklin, Imbert, Abebe, and Mejia-Mantilla.*

0.1 Setup

Make sure we have the right Working Directory and Julia Environment, change string in `cd` to your own working directory and point `ENV["R_HOME"] = "..."` to your own R home directory.

[ ]:
```
#| echo: true
cd("/Users/bojs/Desktop/Development/replication_public")

using Pkg
Pkg.activate("DevRepPkg")
Pkg.instantiate()
ENV["R_HOME"] = "/Library/Frameworks/R.framework/Resources"
Pkg.build("RCall")
```

Load Necessary Packages

[ ]:
```
#| echo: true
using DataFrames
using Chain
using ReadStatTables
using FixedEffectModels
using Statistics
using GLMNet
```

```
using RegressionTables
using Base.Threads
using StatsBase
using PrettyTables
using Plots
using StatsPlots
using LaTeXStrings
using RCall
```

[ ]:
```
#| echo: true
R"library(sandwich)"
R"library(stargazer)"
```

0.2 Load Data

[ ]:
```
#| echo: true
data = DataFrame(readstat("data/w_i.dta"))
data_ind = DataFrame(readstat("data/ind_i.dta"))
data_hh = @chain begin DataFrame(readstat("data/Y_h.dta"))
    subset(:endline => ByRow(==(1)))
end

T_i = select(data, :selected, :subcity, :new_woreda)
X_i = select(data, :new_woreda, r"^B_C_.*$")
```

[ ]:
```
#| echo: true
pi_i_j = DataFrame(readstat("data/pi_i_j.dta"))

pi_i_j_diagonal = @chain pi_i_j begin
    subset([:O_woreda, :D_woreda] => ByRow((x, y) -> x == y))
    select(Not(r"distance", r"cost", r"time", r"pi"))
end

data_origin = transform(pi_i_j_diagonal,
    :O_selected => :selected,
    :O_FE_commute_out => ByRow(log) => :O_FE_ln_commute_out,
    :O_B_commute_out => ByRow(log) => :O_B_ln_commute_out,
    :D_FE_commute_in => ByRow(log) => :D_FE_ln_commute_in,
    :D_B_commute_in => ByRow(log) => :D_B_ln_commute_in)
```

[ ]:
```
#| echo: true
RI_treatment = @chain DataFrame(readstat("data/rerandomisations_treatment.
↪dta")) begin
    select(:new_woreda, r"^selected_potential.*$")
end
```

```
RI_exposure = @chain DataFrame(readstat("data/rerandomisations_exposure.dta"))␣
↪begin
    select(:D_woreda, r"^D_exposure.*$")
end
```

Amenities Data

[ ]:
```
#| echo: true
B_i = DataFrame(readstat("data/B_i.dta"))

FE_B_i = @chain B_i begin
    select(r"^FE.*$", :selected, :subcity, :new_woreda)
    dropmissing([:FE_B_i, :FE_ln_r_i])
end

SE_B_i = @chain B_i begin
    select(r"^SE.*$")
    dropmissing([:SE_ln_r_i, :SE_B_i])
end

B_i_hh = DataFrame(readstat("data/B_i_hh.dta"))

FE_B_i_hh = @chain B_i_hh begin
    select(r"^FE.*$",
    :selected, :subcity, :new_woreda,
    :ben_pw_select, :spill_select, :eligible_pw)
    dropmissing([:FE_B_i, :FE_ln_r_i])
end

SE_B_i_hh = @chain B_i_hh begin
    select(r"^SE.*$")
    dropmissing([:SE_ln_r_i, :SE_B_i])
end

data_ind = innerjoin(data_ind, DataFrame(
    O_woreda = data_origin.O_woreda,
    O_exposure_sq_rec = data_origin.O_exposure_sq_rec,
    D_exposure_noi=data_origin.D_exposure_noi,
    D_exposure_noi_rex=data_origin.D_exposure_noi_rec
), on = :new_woreda => :O_woreda, makeunique = true)
```

0.3 T2 Treatment Effects on Labour Outcomes

I now replicate the results on the direct effects of the publi cowrks program on labor markets using the specification:

$$Y_{i1} = \alpha Y_{i0} + \beta T_i + \delta X_i + \varepsilon_i$$

[ ]:

```
controls = @chain data begin
    select(r"^B_C_.*$")
    names()
    Symbol.()
    Term.()
end

r_work_ind = reg(data_ind,
    Term(:FE_share_hours_work)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_work_mean_ind = @chain data begin
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_work => mean)
    first()
    first()
    round(;digits=3)
    string()
end

r_pw_ind = reg(data_ind,
    Term(:FE_share_hours_pw)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_pw_mean_ind = @chain data begin
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_pw => mean)
    first()
    first()
    round(;digits=3)
    string()
end

r_nonpw_ind = reg(data_ind,
    Term(:FE_share_hours_nonpw)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_nonpw_mean_ind = @chain data begin
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_nonpw => mean)
    first()
```

```
    first()
    round(;digits=3)
    string()
end
```

[ ]:

```
# Eligible
r_work_el = reg(subset(data_ind, :eligible_pw => ByRow(==(1))),
    Term(:FE_share_hours_work)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_work_mean_el = @chain data_ind begin
    subset(:eligible_pw => ByRow(==(1)))
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_work => mean)
    first()
    first()
    round(;digits=3)
    string()
end

r_pw_el = reg(subset(data_ind, :eligible_pw => ByRow(==(1))),
    Term(:FE_share_hours_pw)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_pw_mean_el = @chain data_ind begin
    subset(:eligible_pw => ByRow(==(1)))
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_pw => mean)
    first()
    first()
    round(;digits=3)
    string()
end

r_nonpw_el = reg(subset(data_ind, :eligible_pw => ByRow(==(1))),
    Term(:FE_share_hours_nonpw)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_nonpw_mean_el = @chain data_ind begin
    subset(:eligible_pw => ByRow(==(1)))
    subset(:selected => ByRow(==(0)))
```

```
    combine(:FE_share_hours_nonpw => mean)
    first()
    first()
    round(;digits=3)
    string()
end

# Ineligible

r_work_inel = reg(subset(data_ind, :eligible_pw => ByRow(==(0))),
    Term(:FE_share_hours_work)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_work_mean_inel = @chain data_ind begin
    subset(:eligible_pw => ByRow(==(0)))
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_work => mean)
    first()
    first()
    round(;digits=3)
    string()
end

r_pw_inel = reg(subset(data_ind, :eligible_pw => ByRow(==(0))),
    Term(:FE_share_hours_pw)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)

FE_share_hours_pw_mean_inel = @chain data_ind begin
    subset(:eligible_pw => ByRow(==(0)))
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_pw => mean)
    first()
    first()
    round(;digits=3)
    string()
end

r_nonpw_inel = reg(subset(data_ind, :eligible_pw => ByRow(==(0))),
    Term(:FE_share_hours_nonpw)
       ~ Term(:selected) + sum(controls) + fe(Term(:subcity)),
    Vcov.cluster(:new_woreda),
    weights = :FE_W_ind_weight)
```

```
FE_share_hours_nonpw_mean_inel = @chain data_ind begin
    subset(:eligible_pw => ByRow(==(0)))
    subset(:selected => ByRow(==(0)))
    combine(:FE_share_hours_nonpw => mean)
    first()
    first()
    round(;digits=3)
    string()
end
```

```
[ ]: FE_controls = @chain FE_B_i_hh begin
    select(r"^FE_c_.*$")
    Matrix()
end

FE_controls_names = @chain FE_B_i_hh begin
    select(r"^FE_c_.*$")
    names()
end

FE_outcome = @chain FE_B_i_hh begin
    select(:FE_B_i)
    Matrix()
    vec()
end

cv_output = glmnetcv(FE_controls, FE_outcome; alpha = 1)
lasso_best = glmnet(FE_controls, FE_outcome, lambda = [lambdamin(cv_output)];␣
 ↪alpha = 1)

selected_indices = findall(!iszero, lasso_best.betas)

#I'm not getting the same thing out here
controls_selected = @chain FE_controls_names[selected_indices] begin
    Symbol.()
    Term.()
end

r_index_quality_ind = reg(B_i_hh, @formula(FE_B_i ~ selected + fe(subcity)),␣
 ↪Vcov.cluster(:new_woreda), weights = :FE_W_hh_weight)

r_index_quality_controls_ind = reg(B_i_hh, Term(:FE_B_i) ~ Term(:selected) +␣
 ↪sum(controls_selected) + fe(Term(:subcity)), Vcov.cluster(:new_woreda),␣
 ↪weights = :FE_W_hh_weight)

# Using the controls that I get out of R
```

```
#r_index_quality_controls_ind = reg(B_i_hh, @formula(FE_B_i ~ selected +␣
↪FE_c__Ihouse_typ_2 + fe(subcity)), weights = :FE_W_hh_weight)

FE_B_i_mean_ind = @chain B_i_hh begin
    subset(:selected => ByRow(==(0)))
    dropmissing(:FE_B_i)
    combine(:FE_B_i => mean)
    first()
    first()
    round(;digits=3)
    string()
end
```

[ ]:
```
## ELIGIBLE/INELIGIBLE
r_index_quality_el = reg(subset(B_i_hh, :eligible_pw => ByRow(==(1))),␣
↪@formula(FE_B_i ~ selected + fe(subcity)), Vcov.cluster(:new_woreda),␣
↪weights = :FE_W_hh_weight)

r_index_quality_inel = reg(subset(B_i_hh, :eligible_pw => ByRow(==(0))),␣
↪@formula(FE_B_i ~ selected + fe(subcity)), Vcov.cluster(:new_woreda),␣
↪weights = :FE_W_hh_weight)

FE_B_i_mean_el = @chain B_i_hh begin
    subset(:eligible_pw => ByRow(==(1)))
    subset(:selected => ByRow(==(0)))
    dropmissing(:FE_B_i)
    combine(:FE_B_i => mean)
    first()
    first()
    round(;digits=3)
    string()
end

FE_B_i_mean_inel = @chain B_i_hh begin
    subset(:eligible_pw => ByRow(==(0)))
    subset(:selected => ByRow(==(0)))
    dropmissing(:FE_B_i)
    combine(:FE_B_i => mean)
    first()
    first()
    round(;digits=3)
    string()
end
```

Table 2a: Full Sample

```
[ ]: #| output: true
# REGRESSION TABLES
#Panel A : All
T2a = regtable(
    r_work_ind, r_pw_ind, r_nonpw_ind, r_index_quality_ind;
    render = HtmlTable(),
    labels = Dict(
        "selected" => "Treatment",
        "FE_share_hours_work" => "Employment",
        "FE_share_hours_pw" => "Public Employment",
        "FE_share_hours_nonpw" => "Private Employment",
        "FE_B_i" => "Neighbourhood Amenities",
        "subcity" => "Subcity"
    ),
    keep = ["Treatment"],
    extralines = [
        ["Control Mean", FE_share_hours_work_mean_ind,␣
 ↪FE_share_hours_pw_mean_ind, FE_share_hours_nonpw_mean_ind, FE_B_i_mean_ind]
    ],
    regression_statistics = [
        Nobs => "Observations"
    ]
)
```

Table 2b: Eligible Households

```
[ ]: #| output: true
# Panel B
T2b = regtable(
    r_work_el, r_pw_el, r_nonpw_el, r_index_quality_el;
    render = HtmlTable(),
    labels = Dict(
        "selected" => "Treatment",
        "FE_share_hours_work" => "Employment",
        "FE_share_hours_pw" => "Public Employment",
        "FE_share_hours_nonpw" => "Private Employment",
        "FE_B_i" => "Neighbourhood Amenities",
        "subcity" => "Subcity"
    ),
    keep = ["Treatment"],
    extralines = [
        ["Control Mean",
        FE_share_hours_work_mean_el, FE_share_hours_pw_mean_el,␣
 ↪FE_share_hours_nonpw_mean_el, FE_B_i_mean_el]
    ],
    regression_statistics = [
        Nobs => "Observations"
```

```
    ]
)
```

Table 2c: Ineligible Households

```
[ ]: #| output: true
# Panel B
T2b = regtable(
    r_work_inel, r_pw_inel, r_nonpw_inel, r_index_quality_inel;
    render = HtmlTable(),
    labels = Dict(
        "selected" => "Treatment",
        "FE_share_hours_work" => "Employment",
        "FE_share_hours_pw" => "Public Employment",
        "FE_share_hours_nonpw" => "Private Employment",
        "FE_B_i" => "Neighbourhood Amenities",
        "subcity" => "Subcity"
    ),
    keep = ["Treatment"],
    extralines = [
        ["Control Mean",
        FE_share_hours_work_mean_el, FE_share_hours_pw_mean_el,␣
 ↪FE_share_hours_nonpw_mean_el, FE_B_i_mean_el]
    ],
    regression_statistics = [
        Nobs => "Observations"
    ]
)
```

0.4 F3 Wages on Exposure Plot

```
[ ]: #| output: true
data_RI = leftjoin(data_origin, RI_exposure, on = :D_woreda)

# Make Scatter Plot
@df data_RI scatter(
    :D_exposure,
    :D_FE_ln_wage,
    group = :D_selected,
    label = ["Control" "Treated"]
)

#Add Line of Best Fit for Control
data_RI0 = subset(data_RI, :D_selected => ByRow(==(0)))
x0 = [ones(length(data_RI0.D_exposure)) data_RI0.D_exposure]
y0 = data_RI0.D_FE_ln_wage
b0 = (x0'x0)\(x0'y0)
```

```
xs0 = data_RI0.D_exposure
ys0 = b0[1] .+ b0[2] .* xs0
plot!(xs0, ys0, seriescolor = "blue", label = false)

# Add Line of Best fit for Treatment
data_RI1 = subset(data_RI, :D_selected => ByRow(==(1)))
x1 = [ones(length(data_RI1.D_exposure)) data_RI1.D_exposure]
y1 = data_RI1.D_FE_ln_wage
b1 = (x1'x1)\(x1'y1)
xs1 = data_RI1.D_exposure
ys1 = b1[1] .+ b1[2] .* xs1
plot!(xs1, ys1, seriescolor = "orange", label = false)

# Add Labels for Axes
xlabel!("Exposure")
ylabel!("Log Wages")
```

0.5 T3 Wage Effects

[ ]:
```
controls = @chain data begin
    select(r"^FE_CW_I_.*$")
    names()
    Symbol.()
    Term.()
end

data_RI = leftjoin(data, RI_treatment; on = :new_woreda, makeunique = true)

data_RI[!, :weight] .= 1

# First Naive Regressions of Wages on Treatment Assignment at Origin without␣
 ↪Controls

r_naive_nocon = reg(data_RI, @formula(FE_ln_earnings_hour_nonpw ~ selected +␣
 ↪B_ln_earnings_hour_nonpw + fe(subcity)),
weights = :weight)

tstats = FixedEffectModels.coef(r_naive_nocon)[1]/sqrt(vcov(r_naive_nocon)[1,1])

tstats_RI = zeros(2000)

function tstatsri(x::Integer)
     RI = reg(data_RI, Term(:FE_ln_earnings_hour_nonpw) ~␣
 ↪Term(Symbol("selected_potential_$x")) + Term(:B_ln_earnings_hour_nonpw) +␣
 ↪fe(Term(:subcity)), weights = :weight, nthreads = 8)
     return FixedEffectModels.coef(RI)[1]/sqrt(vcov(RI)[1,1])
end
```

```
# Get a slight performance boost from doing this in parallel.
# reg() does each regression using multithreading anyway but seems to be␣
↪marginally faster this way.
@fastmath for x in 1:2000
    tstats_RI[x] = tstatsri(x)
end

tstats = append!([tstats], tstats_RI)
tstats = tstats.^2
pvalue_naive_origin_nocon_RI = 1 - quantilerank(tstats, tstats[1])
```

[ ]:
```
# Now with controls
r_naive = reg(data_RI, Term(:FE_ln_earnings_hour_nonpw) ~ Term(:selected) +␣
↪Term(:B_ln_earnings_hour_nonpw) + sum(controls) + fe(Term(:subcity)),␣
↪weights = :weight)

tstats = FixedEffectModels.coef(r_naive)[1]/sqrt(vcov(r_naive)[1,1])

tstats_RI = zeros(2000)

function tstatsri(x::Integer)
    RI = reg(data_RI, Term(:FE_ln_earnings_hour_nonpw) ~␣
↪Term(Symbol("selected_potential_$x")) + Term(:B_ln_earnings_hour_nonpw) +␣
↪sum(controls) + fe(Term(:subcity)), weights = :weight, nthreads = 8)
    return FixedEffectModels.coef(RI)[1]/sqrt(vcov(RI)[1,1])
end

tstats_RI = zeros(2000)

@fastmath for x in 1:2000
    tstats_RI[x] = tstatsri(x)
end

tstats = append!([tstats], tstats_RI)
tstats = tstats.^2
pvalue_naive_origin_RI = 1 - quantilerank(tstats, tstats[1])
```

[ ]:
```
# exposure - No Controls
controls = @chain data_origin begin
    select(r"^D_FE_CC_I_.*$")
    names()
    Symbol.()
    Term.()
end

data_RI = leftjoin(data_origin, RI_exposure; on = :D_woreda, makeunique = true)
data_RI.weight .= 1
```

```
r_spill_nocon = reg(data_RI, @formula(D_FE_ln_wage ~  D_exposure_rec +
 ↪D_B_ln_wage), weights = :weight)
tstats = FixedEffectModels.coef(r_spill_nocon)[1]/vcov(r_spill_nocon)[1,1]


function tstatsri(x::Integer)
    RI = reg(data_RI, Term(:D_FE_ln_wage) ~ Term(Symbol("D_exposure_rec_$x")) +
 ↪Term(:D_B_ln_wage), weights = :weight, nthreads = 8)
    return FixedEffectModels.coef(RI)[1]/sqrt(vcov(RI)[1,1])
end

tstats_RI = zeros(2000)

@fastmath for x in 1:2000
    tstats_RI[x] = tstatsri(x)
end
tstats = append!([tstats], tstats_RI)

tstats = tstats.^2
pvalue_spillovers_nocon_RI = 1 - quantilerank(tstats, tstats[1])
```

[ ]:
```
# exposure - Controls

r_spill = reg(data_RI, Term(:D_FE_ln_wage) ~  Term(:D_exposure_rec) + Term(:
 ↪D_B_ln_wage) + sum(controls), weights = :weight)
tstats = FixedEffectModels.coef(r_spill)[1]/vcov(r_spill)[1,1]

function tstatsri(x::Integer)
    RI = reg(data_RI, Term(:D_FE_ln_wage) ~ Term(Symbol("D_exposure_rec_$x")) +
 ↪Term(:D_B_ln_wage) + sum(controls), weights = :weight, nthreads = 8)
    return FixedEffectModels.coef(RI)[1]/sqrt(vcov(RI)[1,1])
end

tstats_RI = zeros(2000)

@fastmath for x in 1:2000
    tstats_RI[x] = tstatsri(x)
end

tstats = append!([tstats], tstats_RI)

tstats = tstats.^2
pvalue_spillovers_RI = 1 - quantilerank(tstats, tstats[1])
```

```
[ ]: #| output: true
T3 = regtable(
    r_naive_nocon, r_naive, r_spill_nocon,r_spill;
    render = HtmlTable(),
    labels = Dict(
        "FE_ln_earnings_hour_nonpw" => "Log wages at origin",
        "D_FE_ln_wage" => "Log wages at destination",
        "selected" => "Treatment at Origin",
        "D_exposure_rec" => "Exposure of Destination"
    ),
    keep = ["Treatment at Origin", "Exposure of Destination"],
    extralines = [
        ["RI p-values",
        ␣
 ↪pvalue_naive_origin_RI,pvalue_naive_origin_nocon_RI,pvalue_spillovers_nocon_RI,pvalue_spill
        ["Worker Controls", "No", "Yes", "No", "Yes"]
    ],
    regression_statistics = [
        Nobs => "Observations"
    ]
)
```

0.6 T4 Valuing Amenities through correlation with rent.

Estimate Correlation between amenities and rent.

```
[ ]: SE_controls = @chain SE_B_i_hh begin
    select(r"^SE_c_.*$")
    Matrix()
end

SE_ln_r_i = @chain SE_B_i_hh begin
    select(:SE_ln_r_i)
    Matrix()
    vec()
end

cv_output = glmnetcv(SE_controls, SE_ln_r_i; alpha = 1)
lasso_best = glmnet(SE_controls, SE_ln_r_i; alpha = 1, lambda =␣
 ↪[lambdamin(cv_output)])
controls_selected_r_i = Term.(Symbol.(names(select(SE_B_i_hh, r"^SE_c_.
 ↪*$"))[findall(!iszero, lasso_best.betas)]))

r_rent_quality = reg(B_i_hh, @formula(SE_ln_r_i ~ SE_B_i + fe(subcity)), Vcov.
 ↪cluster(:new_woreda))
```

```
r_rent_quality_controls = reg(B_i_hh, Term(:SE_ln_r_i) ~ Term(:SE_B_i) +␣
↪sum(controls_selected_r_i) + fe(Term(:subcity)), Vcov.cluster(:new_woreda))

T4 = regtable(
    r_rent_quality, r_rent_quality_controls,
    render = HtmlTable(),
    labels = Dict(
        "SE_B_i" => "Neighborhood Quality",
        "SE_ln_r_i" => "Log Rent"
    ),
    keep = ["Neighborhood Quality"],
    regression_statistics = [
        Nobs => "Observations"
    ]
)
```

0.7 T5 Gravity

Now regress the commuting probability on the log wage at destination. Leverage the experiment and instrument the log wage at destination with the destination's exposure to treatment. We use origin fixed effects to control for origin level amenities and wages.

$$\ln \pi_{ij} = \theta \ln w_j + \kappa\theta \ln d_{ij} + \nu_i + \varepsilon_{ij}$$

[ ]:
```
controls = @chain pi_i_j begin
    select(r"^D_FE_CC_I_")
    names()
    Symbol.()
    Term.()
end

data_poisson = @chain pi_i_j begin
    dropmissing(:ln_walking_time)
    transform(:O_FE_residents_workers => :weight)
end

@rput data_poisson

R"""
r_wage_poisson <- glm(O_FE_pi_i_j ~ D_FE_ln_wage + ln_walking_time  +␣
↪D_B_ln_wage +  as.factor(O_woreda), data = data_poisson, family =␣
↪poisson(link = 'log'), na.action = na.omit, weights = weight)
r_wage_poisson_se <- sqrt(diag(vcovCL(r_wage_poisson, cluster= ~  D_woreda)))
# This second equation regresses log wages on exposure #
formula <- as.formula((paste("D_FE_ln_wage ~ D_exposure_rec   + ln_walking_time␣
↪ + D_B_ln_wage +  as.factor(O_woreda) ", sep = ' ')))
```

```
r_wage_poisson_fs=lm(formula, data = data_poisson, na.action = na.
 ↪omit,weights=weight)
r_wage_poisson_fs_se <-sqrt(diag(vcovCL(r_wage_poisson_fs, cluster= ~ ␣
 ↪D_woreda)))

# The residuals from that first stage are then used as a control to get IV␣
 ↪estimates (that's the wooldridge method)
data_poisson$control_wage <- residuals(r_wage_poisson_fs)

formula <- as.formula(paste("O_FE_pi_i_j ~ D_FE_ln_wage    + ln_walking_time +␣
 ↪control_wage + D_B_ln_wage  +  as.factor(O_woreda) ", sep = ' ') )
suppressWarnings({
r_wage_poisson_iv=glm(formula, data = data_poisson, family=poisson(link="log"),␣
 ↪na.action = na.omit,weights=weight)
})
r_wage_poisson_iv_se <-sqrt(diag(vcovCL(r_wage_poisson_iv, cluster= ~ ␣
 ↪D_woreda)))
"""
```

[ ]:
```
#/output: true
data_BS= select(data_poisson, :weight,:O_woreda,:D_woreda,:O_FE_pi_i_j,:
 ↪D_FE_ln_wage,:ln_walking_time,:D_B_ln_wage,:D_subcity,:O_FE_residents,:
 ↪D_exposure_rec,r"^D_FE_CC_I_.*$")
origins = select(data_BS, :O_woreda)

@rput data_BS
@rput origins

R"""

origins = as.numeric(unlist(unique(origins)))


beta_BS=matrix(NA,200)
start_time=Sys.time()

for (x in 1:200){
  set.seed(x)
  resample=sample(origins, replace = TRUE)
  O_vector=as.vector(resample)
  O_vector=as.data.frame(O_vector)
  names(O_vector)=c("O_vector")
  data_poisson=merge(data_BS, O_vector, by.x=c("O_woreda"), by.y=c("O_vector"),␣
 ↪all.x=FALSE)

# This second equation regresses log wages on exposure #
```

```
  r_wage_poisson_fs_BS=lm(D_FE_ln_wage ~ D_exposure_rec   + ln_walking_time  +␣
 ↪D_B_ln_wage  +  as.factor(O_woreda), data = data_poisson, na.action = na.
 ↪omit,weights=weight)

# The residuals from that first stage are then used as a control to get IV␣
 ↪estimates (that's the wooldridge method)
data_poisson$control_wage <- residuals(r_wage_poisson_fs_BS)

formula <- as.formula(paste('O_FE_pi_i_j ~ D_FE_ln_wage    + ln_walking_time +␣
 ↪control_wage + D_B_ln_wage  +  as.factor(O_woreda)  ', sep = ' ') )
suppressWarnings({
 r_wage_poisson_iv_BS=glm(formula, data = data_poisson,␣
 ↪family=poisson(link="log"), na.action = na.omit,weights=weight)
 })
 beta_BS[x,]=c(summary(r_wage_poisson_iv_BS)$coefficients[2])
}


 end_time=Sys.time()
 end_time-start_time
 r_wage_poisson_iv_se[2]=sd(beta_BS)

 T5 = stargazer(r_wage_poisson,r_wage_poisson_iv,r_wage_poisson_fs,type="html",␣
 ↪␣
 ↪se=list(r_wage_poisson_se,r_wage_poisson_iv_se,r_wage_poisson_fs_se),report="vcs",␣
 ↪omit=c("B_","FE_CC_","control","Constant","subcity","woreda"),
           covariate.labels   = c("Log Destination Wage","Destination Exposure␣
 ↪to Program","Log walking time"),
           dep.var.labels   = c("Commuting Probability","Log Destination␣
 ↪Wage"),
           omit.stat=c("ser", "adj.rsq", "f", "rsq"),
          notes.append = FALSE, notes="All specifications include origin fixed␣
 ↪effects")

"""

@rget T5
```

0.8 Model Parameters

To paremetrrize the model we use:

- ΔL_i is the change in total employment in the reduced form ITT.
- p is the change in private sector employment in the reduced form ITT.
- L_i is the employment rate in the control.
- The wage premium g is the difference between log earning on public works at endline in

treatment and log earnings on private sector work at baseline.

- The change in the wage at destination is a function of exposure to the treatment:

$$\widehat{w_j} = \delta * \sum_k \pi_{kj} T_k$$

where δ is the coefficient on exposure in the previous estimation.

- β_i is the product between the increase in neighborhood quality and the correlation between neighborhood quality and rents.
- θ comes from the gravity equation.

Note that `FE_W_woreda_weight` variable does not appear to be available in the provided dataset, so non-individual means are unweighted (unless the variables have been pre-weighted or something).

```
[ ]: #| output: true
ln_earnings_means_ind = @chain data_ind begin
    dropmissing([:FE_ln_earnings_hour_pw, :FE_ln_earnings_hour_nonpw])
    subset(:selected => ByRow(==(1)))
    DataFrames.combine([:FE_ln_earnings_hour_nonpw, :FE_W_ind_weight] => (x, y)␣
 ↪-> mean(x, weights(y)),
            [:FE_ln_earnings_hour_pw, :FE_W_ind_weight] => (x, y) -> mean(x,␣
 ↪weights(y)))
end

ln_earnings_means = @chain data begin
    subset(:selected => ByRow(==(1)))
    DataFrames.combine(
        :FE_ln_earnings_hour_pw => mean,
        :B_ln_earnings_hour_nonpw => mean
    )
end

w_g = first(ln_earnings_means[!, 1])

data_g = @chain data begin
    select(:new_woreda, :B_ln_earnings_hour_nonpw)
    transform(:B_ln_earnings_hour_nonpw => (x -> w_g .- x )=> :g)
    select(:new_woreda, :g)
end

R"theta = coef(r_wage_poisson_iv)[2]"

@rget theta

Delta_L_i = FixedEffectModels.coef(r_work_ind)[1]
```

```
p = -FixedEffectModels.coef(r_nonpw_ind)[1]/parse(Float64,␣
↪FE_share_hours_work_mean_ind)
L_i = parse(Float64, FE_share_hours_work_mean_ind)
wage_effect = FixedEffectModels.coef(r_spill)[2]

beta_i = FixedEffectModels.coef(r_rent_quality_controls)[1]*FixedEffectModels.
↪coef(r_index_quality_controls_ind)[1]

wage_effect_naive = FixedEffectModels.coef(r_naive)[1]

params = Dict(
    L"w_g" => w_g,
    L"\theta" => theta,
    L"L_i" => L_i,
    L"\Delta_L_i" => Delta_L_i,
    L"p" => p,
    "Wage Effect" => wage_effect,
    L"/beta_i" => beta_i
)

pretty_table(params, backend = Val(:html))
```

0.9 T6 Welfare

From the model, the expression of changes in welfare is the following:

$$\widehat{U_i} = (1+\beta_i)\left[pT_i(1+g_i)\pi_{ii}^{\frac{1}{\theta}} + (1-pT_i)\left(\sum_j \pi_{ij}(\widehat{w_j})^{\theta}\right)^{\frac{1}{\theta}}\right]$$

And we can benchmark the welfare effects of the program with a cash transfer.

$$\widehat{U_i^{cash}} = \left[1 + p(1+g_i)\pi_{ii}^{\frac{1}{\theta}}\right]$$

[ ]:
```
#| output: true
woreda_wage_dest_effect = @chain pi_i_j begin
    transform(:D_exposure => (x -> 1 .+ wage_effect .* x) => :
↪woreda_wage_dest_effect)
    groupby(:D_woreda)
    combine(:woreda_wage_dest_effect => mean)
end

data_welfare = innerjoin(data_origin, woreda_wage_dest_effect, on = :D_woreda)

sum_wage_effects = @chain pi_i_j begin
    transform([:O_B_pi_i_j, :D_exposure] => ((x, y) -> (x .* (1 .+ wage_effect .
↪* y).^theta)) => :sum_wage_effects)
```

```
    groupby(:O_woreda)
    combine(
        :sum_wage_effects => sum
    )
end

data_welfare = innerjoin(data_welfare, sum_wage_effects, on = :O_woreda)

data_welfare = innerjoin(data_welfare, data_g, on = [:O_woreda => :new_woreda])

# partial roll out
f1 = (s, c, w, g) -> s*(1+beta_i)*(p*(1+g)*(1-c)^(1/theta)+(1 - p)*w^(1/
 ↪theta))+(1-s)*w^(1/theta)
f2 = (s, c, w, g) -> s*(p*(1+g)*(1-c)^(1/theta)+(1-p)*(w)^(1/
 ↪theta))+(1-s)*(w)^(1/theta)
f3 = (s, c, g) -> s*((1-p)+p*(1+g)*(1-c)^(1/theta))+(1-s)
f4 = (s, c, g) -> s*(1+beta_i)*((1-p)+p*(1+g)*(1-c)^(1/theta))+(1-s)
f5 = (s, c, g) -> s*(1+p*(1+g)*(1-c)^(1/theta))+(1-s)


transform!(
    data_welfare,
    [:O_selected, :O_B_commute_out, :sum_wage_effects_sum, :g] => ByRow((s, c,␣
 ↪w, g) -> f1(s, c, w, g)) => :u_i,
    [:O_selected, :O_B_commute_out, :sum_wage_effects_sum, :g] => ByRow((s, c,␣
 ↪w, g) -> f2(s, c, w, g)) => :u_i_no_amenity,
    [:O_selected, :O_B_commute_out, :g] => ByRow((s, c, g) -> f3(s, c, g)) => :
 ↪u_i_no_wage_no_amenity,
    [:O_selected, :O_B_commute_out, :g] => ByRow((s, c, g) -> f4(s, c, g)) => :
 ↪u_i_no_wage,
    [:O_selected, :O_B_commute_out, :g] => ByRow((s, c, g) -> f5(s, c, g)) => :
 ↪u_i_cash
)

# full roll out
sum_wage_effects_full = @chain pi_i_j begin
    transform(:O_B_pi_i_j => (x -> x .* (1 + wage_effect)^theta) => :
 ↪wage_effects_full)
    groupby(:O_woreda)
    combine(
        :wage_effects_full => sum
    )
end

data_welfare = innerjoin(data_welfare, sum_wage_effects_full, on = :O_woreda)
data_welfare.O_exposure_full .= 1
```

```julia
data_welfare.O_selected_full .= 1

transform!(
    data_welfare,
    [:O_selected_full, :O_B_commute_out, :wage_effects_full_sum, :g] => ByRow((s, c, w, g) -> f1(s, c, w, g)) => :u_i_full,
    [:O_selected_full, :O_B_commute_out, :wage_effects_full_sum, :g] => ByRow((s, c, w, g) -> f2(s, c, w, g)) => :u_i_full_no_amenity,
    [:O_selected_full, :O_B_commute_out, :g] => ByRow((s, c, g) -> f3(s, c, g)) => :u_i_full_no_wage_no_amenity,
    [:O_selected_full, :O_B_commute_out, :g] => ByRow((s, c, g) -> f4(s, c, g)) => :u_i_full_no_wage,
    [:O_selected_full, :O_B_commute_out, :g] => ByRow((s, c, g) -> f5(s, c, g)) => :u_i_full_cash,
    :wage_effects_full_sum => ByRow(x -> p+(1-p)*(x)^(1/theta)) => :u_i_full_wage_only
)

# Create Table

function rmsuffix(df::DataFrame, n::Int)
    new_names = [first(names(df)[i], length(names(df)[i]) - n) for i in 2:length(names(df))]
    pushfirst!(new_names, names(df)[1])
    rename!(df, Symbol.(new_names))
    return df
end

table_by_selected = @chain data_welfare begin
    groupby(:O_selected)
    combine(
        [
            [:D_exposure, :O_W_woreda_weight],
            [:u_i_no_wage_no_amenity, :O_W_woreda_weight],
            [:u_i_no_amenity, :O_W_woreda_weight],
            [:u_i, :O_W_woreda_weight],
            [:u_i_cash, :O_W_woreda_weight]
        ] .=> (x, y) -> mean(x, weights(y)),
        renamecols = false
    )
    rmsuffix(18)
    transform(
        [:u_i_no_wage_no_amenity,
        :u_i_no_amenity,
        :u_i,
        :u_i_cash] .=> (x -> x .- 1);
```

```
        renamecols = false
    )
end

table_full = @chain data_welfare begin
    combine(
        [
            [:O_selected_full, :O_W_woreda_weight],
            [:O_exposure_full, :O_W_woreda_weight],
            [:u_i_full_no_wage_no_amenity, :O_W_woreda_weight],
            [:u_i_full_no_amenity, :O_W_woreda_weight],
            [:u_i_full, :O_W_woreda_weight],
            [:u_i_full_cash, :O_W_woreda_weight]
        ] .=> (x, y) -> mean(x, weights(y));
        renamecols = false
    )
    rmsuffix(18)
    transform(
        [
        :u_i_full_no_wage_no_amenity,
        :u_i_full_no_amenity,
        :u_i_full,
        :u_i_full_cash
        ] .=> (x -> x .- 1);
        renamecols = false
    )
end

rollout = ["Treatment", "Exposure", "Direct Effect", "Direct + Wage Effects",
 ↪"Direct + Wage + Amenity", "Cash Transfer"]

t6 = hcat(rollout, round.(Matrix(table_by_selected), digits = 3)', round.
 ↪(Matrix(table_full), digits = 3)')

T6 = pretty_table(t6, header = ["Roll-out", "Control", "Treatment", "All"],
 ↪backend = Val(:html))
```