Hierarchical model with risk level intercept Laura Balasso 1/3/2021 Hierarchical model for italian regions regions <- unique(data_it\$region)</pre> regions ## [1] "Abruzzo" "Basilicata" "Calabria" "Emilia-Romagna" "Friuli Venezia Giulia" ## [4] "Campania" ## [7] "Lazio" "Liguria" "Lombardia" ## [10] "Marche" "Molise" "P.A. Bolzano" ## [13] "P.A. Trento" "Piemonte" "Puglia" "Sicilia" "Toscana" ## [16] "Sardegna" ## [19] "Umbria" "Valle d'Aosta" "Veneto" hier_data <- get_hier_data(data_it, regions, Y_delay, O_delay, R_delay, initial_date = as.Date('2020-10-15'), fin $al_date = as.Date('2020-12-22'))$ risk_idx <- matrix(data = NA, nrow = nrow(hier_data\$yellow_dummies), ncol = ncol(hier_data\$yellow_dummies))</pre> for(i in 1:nrow(risk_idx)){ for(j in 1:ncol(risk_idx)){ if(hier_data\$yellow_dummies[i,j] == 1) risk_idx[i, j] <- 2</pre> else if(hier_data\$orange_dummies[i,j] == 1) risk_idx[i,j] <- 3</pre> else if(hier_data\$red_dummies[i,j]==1) risk_idx[i,j] <- 4</pre> risk_idx[is.na(risk_idx)] <- 1</pre> p_delay <- get_delay_distribution()</pre> stan_data_hier <- list(J = length(regions),</pre> N = nrow(hier_data\$exposures), N_nonzero = length(hier_data\$nonzero_days), nonzero_days = hier_data\$nonzero_days, conv_gt = get_gt_convolution_ln2(nrow(hier_data\$exposures)), length_delay = length(p_delay), $p_{delay} = p_{delay}$ exposures = hier_data\$exposures, nonzero_positives = hier_data\$positives[hier_data\$nonzero_days ,], risk_idx = risk_idx[hier_data\$nonzero_days ,] compiled_hier <- stan_model('../stan/hier_area_intercepts.stan')</pre> fit_hier <- sampling(compiled_hier, data = stan_data_hier, iter= 2000)</pre> print(fit_hier, pars='tau') ## Inference for Stan model: hier_area_intercepts. ## 4 chains, each with iter=2000; warmup=1000; thin=1; ## post-warmup draws per chain=1000, total post-warmup draws=4000. ## mean se_mean sd 2.5% 25% 50% 75% 97.5% n_eff Rhat ## tau 0.32 0.03 1.74 0.01 0.03 0.09 0.3 1.77 3772 1 ## Samples were drawn using NUTS(diag_e) at Wed Jan 06 12:41:26 2021. ## For each parameter, n_eff is a crude measure of effective sample size, ## and Rhat is the potential scale reduction factor on split chains (at ## convergence, Rhat=1). mcmc_hist(fit_hier, pars='tau') ## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`. 25 50 75 100 tau print(fit_hier, pars = 'alpha') ## Inference for Stan model: hier_area_intercepts. ## 4 chains, each with iter=2000; warmup=1000; thin=1; ## post-warmup draws per chain=1000, total post-warmup draws=4000. ## mean se_mean sd 2.5% 25% 50% 75% 97.5% n_eff Rhat ## alpha -0.01 0 0 -0.01 -0.01 -0.01 -0.01 -0.01 3139 ## Samples were drawn using NUTS(diag_e) at Wed Jan 06 12:41:26 2021. ## For each parameter, n_eff is a crude measure of effective sample size, ## and Rhat is the potential scale reduction factor on split chains (at ## convergence, Rhat=1). mcmc_hist(fit_hier, pars = 'alpha') ## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`. -0.012 -0.010 alpha Risk level intercepts print(fit_hier, pars = 'risk_level') ## Inference for Stan model: hier_area_intercepts. ## 4 chains, each with iter=2000; warmup=1000; thin=1; ## post-warmup draws per chain=1000, total post-warmup draws=4000. ## mean se_mean sd 2.5% 25% 50% 75% 97.5% n_eff Rhat ## risk_level[1] 18.03 0.05 3.24 11.59 15.89 18.00 20.24 24.40 4296 ## risk_level[2] 32.89 0.09 4.79 23.60 29.54 32.89 36.20 42.25 2910 ## risk_level[3] 29.43 0.08 4.74 20.13 26.12 29.44 32.58 38.87 3322 0.08 3.10 8.17 14.04 16.02 17.70 20.51 1604 ## risk_level[4] 15.61 ## Samples were drawn using NUTS(diag_e) at Wed Jan 06 12:41:26 2021. ## For each parameter, n_eff is a crude measure of effective sample size, ## and Rhat is the potential scale reduction factor on split chains (at ## convergence, Rhat=1). mcmc_hist(fit_hier, pars = c('risk_level[1]', 'risk_level[2]')) ## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`. risk_level[1] risk_level[2] 15 20 25 mcmc_hist(fit_hier, pars = c('risk_level[3]', 'risk_level[4]')) ## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`. risk_level[3] risk_level[4] Trace plots mcmc_trace(as.array(fit_hier, pars = c('tau')), np = nuts_params(fit_hier) ## No divergences to plot. 100 Chain **tan** 50 ___ 2 200 400 800 1000 mcmc_trace(as.array(fit_hier, pars = c('phi')), np = nuts_params(fit_hier) ## No divergences to plot. 22 21 Chain iqd 20 19 18 200 800 1000 Posterior predictive check y_rep <- as.matrix(fit_hier, pars = "y_rep")</pre> ppc_dens_overlay(y = as.vector(stan_data_hier\$nonzero_positives), y_rep[1:1000,]) 15000 10000 20000 Posterior predictive check by region regional_yrep_idx <- function(region, regions_vector, nonzero_days){</pre> region_idx <- which(regions_vector == region)</pre> yrep_idx <- (region_idx-1)* length(nonzero_days) + 1</pre> range <- yrep_idx : (yrep_idx + length(nonzero_days)-1)</pre> return(range) groups <- function(regions, nonzero_days){</pre> group <- rep(regions[1], length(nonzero_days))</pre> for(r in 2:length(regions)) group <- c(group, rep(regions[r], length(nonzero_days)))</pre> return(group) Lombardia $ppc_dens_overlay(y = stan_data_hier nonzero_positives[, which(regions == 'Lombardia')], y_rep[1:1000, regional_yre]$ p_idx('Lombardia', regions, stan_data_hier\$nonzero_days)]) 5000 10000 15000 20000 $ppc_intervals(y = stan_data_hier\$nonzero_positives[, which(regions == 'Lombardia')], y_rep[1:1000, regional_yrep_i = 'Lombardia'], y_rep[1:1000, regional_yrep$ dx('Lombardia', regions, stan_data_hier\$nonzero_days)]) 15000 10000 20 40 Veneto ppc_dens_overlay(y = stan_data_hier\$nonzero_positives[, which(regions == 'Veneto')], y_rep[1:1000,regional_yrep_i dx('Veneto', regions, stan_data_hier\$nonzero_days)]) 9000 12000 3000 6000 ppc_intervals(y = stan_data_hier\$nonzero_positives[, which(regions == 'Veneto')], y_rep[1:1000,regional_yrep_idx('Veneto', regions, stan_data_hier\$nonzero_days)]) 8000 6000 4000 \boldsymbol{x} Emilia Romagna ppc_dens_overlay(y = stan_data_hier\$nonzero_positives[, which(regions == 'Emilia-Romagna')], y_rep[1:1000, regiona l_yrep_idx('Emilia-Romagna', regions, stan_data_hier\$nonzero_days)]) 1000 2000 3000 4000 5000 6000 $ppc_intervals(y = stan_data_hier$nonzero_positives[, which(regions == 'Emilia-Romagna')], y_rep[1:1000, regional_y])$ rep_idx('Emilia-Romagna', regions, stan_data_hier\$nonzero_days)]) 4000 3000 2000 1000 20 60 40 \boldsymbol{x} ppc_stat_grouped(y=as.vector(stan_data_hier\$nonzero_positives), yrep =y_rep, group = groups(regions, stan_data_hi er\$nonzero_days) ,stat="mean", binwidth=0.5) Basilicata Calabria Abruzzo Campania milia-Romagn 400 450 120130140150 240260280300320 200@20@40@600 $16\mathbf{0}\mathbf{7}\mathbf{0}\mathbf{8}\mathbf{0}\mathbf{9}\mathbf{20}\mathbf{2}\mathbf{0}\mathbf{0}0$ ıli Venezia Giı Lazio Liguria Lombardia Marche 500550600650700 500550600650700 400 450 500 180020002200 $450 \\ \mathbf{5000005} \\ 50 \\ \mathbf{6000} \\ 000$ T = meanP.A. Trento P.A. Bolzano Molise Piemonte Puglia $T(y_{\rm rep})$ T(y)70 75 80 85 90 300325350375400 180 200 220 240 1900100300500 100011001200Sicilia Toscana Valle d'Aosta Sardegna Umbria 300325350375 32\$5\$7\$40\$425 100**0**10**0**20**0**300 130D40D50D600 707580859095 Veneto 240**0**60**0**80**0**00**0**200 mean_inv_phi<-mean(rstan::extract(fit_hier)\$inv_phi)</pre> mean_y_rep<-colMeans(y_rep)</pre> $std_resid <- (as.vector(stan_data_hier\$nonzero_positives) - mean_y_rep)/sqrt(mean_y_rep+mean_y_rep^2*mean_inv_phi)$ qplot(mean_y_rep, std_resid)+hline_at(2)+hline_at(-2) 5.0 std_resid 0.0 -6000 3000 9000 12000 mean_y_rep Rt Lombardia plot_rt_hier(hier_data, fit_hier, regions, 'Lombardia') 1.5 -1.0 ott 15 dic 15 nov 01 nov 15 dic 01 Date Rt Veneto plot_rt_hier(hier_data, fit_hier, regions, 'Veneto') 1.50 -1.25 -1.00 -0.75 nov 15 dic 01 dic 15 ott 15 nov 01 Date Rt Emilia Romagna plot_rt_hier(hier_data, fit_hier, regions, 'Emilia-Romagna') 1.4 -1.2 -1.0 - -0.8 -0.6 ott 15 nov 01 nov 15 dic 01 dic 15 Date Looic and WAIC

log_lik <- extract_log_lik(fit_hier)</pre>

loo <- loo(log_lik)
waic <- waic(log_lik)</pre>

loo\$estimates[3,1]

waic\$esestimates[3,1]

[1] 17800.55

NULL