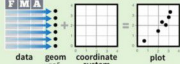


Data Visualization with ggplot2 Cheat Sheet

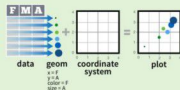


Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

qplot(*x* = *cty*, *y* = *hwy*, *color* = *cyl*, *data* = *mpg*, *geom* = "point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(*data* = *mpg*, *aes*(*x* = *cty*, *y* = *hwy*))

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

ggplot(*mpg*, *aes*(*hwy*, *cty*)) +
geom_point(*aes*(*color* = *cyl*)) +
geom_smooth(*method* = "lm") +
coord_cartesian() +
scale_color_gradient() +
theme_bw()

Add a new layer to a plot with a **geom_***() or **stat_***() function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()

Returns the last plot

ggsave("plot.png", width = 5, height = 5)

Saves last plot as 5 x 5 file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- **ggplot**(*mpg*, *aes*(*hwy*))



a + **geom_area**(*stat* = "bin")
x, *y*, alpha, color, fill, linetype, size
b + **geom_area**(*aes*(*y* = ..density..), *stat* = "bin")
a + **geom_density**(*kernel* = "gaussian")
x, *y*, alpha, color, fill, linetype, size, weight
b + **geom_density**(*aes*(*y* = ..county..))



a + **geom_dotplot**()
x, *y*, alpha, color, fill
a + **geom_freqpoly**()
x, *y*, alpha, color, linetype, size
b + **geom_freqpoly**(*aes*(*y* = ..density..))
a + **geom_histogram**(*binwidth* = 5)
x, *y*, alpha, color, fill, linetype, size, weight
b + **geom_histogram**(*aes*(*y* = ..density..))



Discrete

b <- **ggplot**(*mpg*, *aes*(*fl*))



b + **geom_bar**()
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- **ggplot**(*map*, *aes*(*long*, *lat*))



c + **geom_polygon**(*aes*(*group* = *group*))
x, *y*, alpha, color, fill, linetype, size

d <- **ggplot**(*economics*, *aes*(*date*, *unemploy*))



d + **geom_path**(*lineend* = "butt",
linejoin = "round", *linemitre* = 1)
x, *y*, alpha, color, linetype, size
d + **geom_ribbon**(*aes*(*ymin* = *unemploy* - 900,
ymax = *unemploy* + 900))
x, *ymax*, *ymin*, alpha, color, fill, linetype, size



e <- **ggplot**(*seals*, *aes*(*x* = *long*, *y* = *lat*))



e + **geom_segment**(*aes*(
xend = *long* + *delta_long*,
yend = *lat* + *delta_lat*)
x, *yend*, *yend*, alpha, color, linetype, size



e + **geom_rect**(*aes*(*xmin* = *long*, *ymin* = *lat*,
xmax = *long* + *delta_long*,
ymax = *lat* + *delta_lat*)
xmax, *xmin*, *ymax*, *ymin*, alpha, color, fill,
linetype, size

Two Variables

Continuous X, Continuous Y

f <- **ggplot**(*mpg*, *aes*(*cty*, *hwy*))



f + **geom_blank**()



f + **geom_jitter**()
x, *y*, alpha, color, fill, shape, size



f + **geom_point**()
x, *y*, alpha, color, fill, shape, size



f + **geom_quantile**()
x, *y*, alpha, color, linetype, size, weight



f + **geom_rug**(*sides* = "bl")
alpha, color, linetype, size



f + **geom_smooth**(*model* = *lm*)
x, *y*, alpha, color, fill, linetype, size, weight



f + **geom_text**(*aes*(*label* = *cty*))
x, *y*, label, alpha, angle, color, family, fontface,
hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- **ggplot**(*mpg*, *aes*(*class*, *hwy*))



g + **geom_bar**(*stat* = "identity")
x, *y*, alpha, color, fill, linetype, size, weight



g + **geom_boxplot**()
lower, middle, upper, *x*, *ymax*, *ymin*, alpha,
color, fill, linetype, shape, size, weight



g + **geom_dotplot**(*binaxis* = "y",
stackdir = "center")
x, *y*, alpha, color, fill



g + **geom_violin**(*scale* = "area")
x, *y*, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- **ggplot**(*diamonds*, *aes*(*cut*, *color*))



h + **geom_jitter**()
x, *y*, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- **ggplot**(*movies*, *aes*(*year*, *rating*))



i + **geom_bin2d**(*binwidth* = *c*(5, 0.5))
xmax, *xmin*, *ymax*, *ymin*, alpha, color, fill,
linetype, size, weight



i + **geom_density2d**()
x, *y*, alpha, colour, linetype, size



i + **geom_hex**()
x, *y*, alpha, colour, fill size

Continuous Function

j <- **ggplot**(*economics*, *aes*(*date*, *unemploy*))



j + **geom_area**()
x, *y*, alpha, color, fill, linetype, size



j + **geom_line**()
x, *y*, alpha, color, linetype, size



j + **geom_step**(*direction* = "hv")
x, *y*, alpha, color, linetype, size

Visualizing error

df <- **data.frame**(*grp* = *c*("A", "B"), *fit* = 4.5, *se* = 1.2)
k <- **ggplot**(*df*, *aes*(*grp*, *fit*, *ymin* = *fit* - *se*, *ymax* = *fit* + *se*))



k + **geom_crossbar**(*latten* = 2)
x, *y*, *ymax*, *ymin*, alpha, color, fill, linetype,
size



k + **geom_errorbar**()
x, *ymax*, *ymin*, alpha, color, linetype, size,
width (also **geom_errorbarh**!)



k + **geom_linerange**()
x, *ymin*, *ymax*, alpha, color, linetype, size



k + **geom_pointrange**()
x, *y*, *ymin*, *ymax*, alpha, color, fill, linetype,
shape, size

Maps

data <- **data.frame**(*murder* = *USArrests\$Murder*,
state = *tolower*(*rownames*(*USArrests*)))
m <- **map_data**("state")
l <- **ggplot**(*data*, *aes*(*fill* = *murder*))



l + **geom_map**(*aes*(*map_id* = *state*), *map* = *map*) +
expand_limits(*x* = *map\$long*, *y* = *map\$lat*)
map_id, alpha, color, fill, linetype, size

Three Variables

seals\$z <- **with**(*seals*, **sqrt**(*delta_long**2 + *delta_lat**2))
m <- **ggplot**(*seals*, *aes*(*long*, *lat*))



m + **geom_contour**(*aes*(*z* = *z*))
x, *y*, *z*, alpha, colour, linetype, size, weight



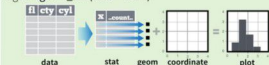
m + **geom_raster**(*aes*(*fill* = *z*), *hjust* = 0.5,
vjust = 0.5, *interpolate* = FALSE)
x, *y*, alpha, fill



m + **geom_title**(*aes*(*fill* = *z*))
x, *y*, alpha, color, fill, linetype, size

Stats - An alternative way to build a layer

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. **a + geom_bar(stat="bin")**



Each stat creates additional variables to map aesthetics to these. These variables use a common **name**, **syntax**.

stat functions and geom functions both combine a stat with a geom to make a layer, i.e. **stat_bin(geom="bar")** does the same as **geom_bar(stat="bin")**

stat function layer specific mappings variable created by transformation

i + stat_density2d(aes(fill = ..level..), geom = "polygon", n = 100)
geom for layer parameters for stat

a + stat_bin(binswidth = 1, origin = 0) 1D distributions
x.y | ..count.., ..density.., ..ndensity..
a + stat_bin2d(binswidth = 1, binsize = "x")
x.y | ..count.., ..density..
a + stat_density(adjust = 1, kernel = "gaussian")
x.y | ..count.., ..density.., ..scaled..

f + stat_bin2d(bins = 30, drop = TRUE) 2D distributions
x.y, fill | ..count.., ..density..
a + stat_bin2d(bins = 30, drop = TRUE)
x.y, fill | ..count.., ..density..
f + stat_density2d(contour = TRUE, n = 100)
x.y, color, size | ..level..

m + stat_contour(aes(z = z)) 3 Variables
x.y, z, order | ..level..
m + stat_spline(aes(radius = z, angle = z))
angle, radius, x, yend, yend | ..xend.., ..yend..
m + stat_summary_hex(aes(z = z), bins = 30, fun = mean)
x.y, z, fill | ..value..
m + stat_summary2d(aes(z = z), bins = 30, fun = mean)
x.y, z, fill | ..value..

g + stat_boxplot(coef = 1.5) Comparisons
x.y | ..lower.., ..middle.., ..upper.., ..outliers..
g + stat_ydensity(adjust = 1, kernel = "gaussian", scale = "area")
x.y | ..density.., ..scaled.., ..count.., ..n.., ..volumwidth.., ..width..

f + stat_ecdf(n = 40) Functions
x.y | ..x.., ..y..
f + stat_quantile(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "n")
x.y | ..quantile.., ..x.., ..y..
f + stat_smooth(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)
x.y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd = 0.5)) General Purpose
x | ..y..

f + stat_identity()
ggplot() + stat_qq(aes(sample = 1:100), distribution = qt, dparams = list(df = 50))
sample, x.y | ..x.., ..y..
f + stat_sum()
x.y | ..size..
f + stat_summary(fun.data = "mean_cl_boot")
f + stat_unique()

Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.



n <- b + geom_bar(aes(fill = f))
scale_fill_manual(values = c("skyblue", "royalblue", "blue", "navy"), limits = c("d", "e", "p", "r"), breaks = c("d", "e", "p", "r"), name = "fuel", labels = c("D", "E", "P", "R"))
range of values to include in mapping **title to use in legend axis** **labels to use in legend axis** **breaks to use in legend axis**
General Purpose scales
Use with any aesthetic:
alpha, color, fill, linetype, shape, size
scale_*_continuous() - map cont' values to visual values
scale_*_discrete() - map discrete values to visual values
scale_*_identity() - use data values as visual values
scale_*_manual(values = c()) - map discrete values to manually chosen visual values

X and Y location scales
Use with x or y aesthetics (x shown here)
scale_x_date(labels = date_format("%%m/%d"), breaks = date_breaks("2 weeks")) - treat x values as dates. Use %strptime for label formats.
scale_x_datetime() - treat x values as date times. Use same arguments as scale_x_date().
scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of a x axis
scale_x_sqrt() - Plot x on square root scale

Color and fill scales
Discrete
n <- b + geom_bar(aes(fill = f))
aes(fill = f)
scale_fill_brewer(palette = "Blues")
For palette choices: library(RColorBrewer); display.brewer.all()
n <- b + geom_bar(aes(fill = f))
scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")
Continuous
n <- a + geom_dotplot(aes(fill = ..x..))
scale_fill_gradient(palette = "Blues")
low = "red", high = "yellow"
scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25)
scale_fill_gradient(colors = terrain.colors(5))
Also: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer: brewer.pal()

Shape scales
n <- f + geom_point(aes(shape = f))
scale_shape(solid = FALSE)
scale_shape_manual(values = c(3, 7))
Shapes values shown in chart on right
Manual shape values
0 6 12 18 24
1 7 8 13 16 20 25
2 8 9 14 15 19 21
3 9 10 15 16 20 21
4 10 11 16 17 21 22
5 11 12 17 18 22 23
6 12 13 18 19 23 24
7 13 14 19 20 24 25
8 14 15 20 21 25 26
9 15 16 21 22 26 27
10 16 17 22 23 27 28
11 17 18 23 24 28 29
12 18 19 24 25 29 30
13 19 20 25 26 30 31
14 20 21 26 27 31 32
15 21 22 27 28 32 33
16 22 23 28 29 33 34
17 23 24 29 30 34 35
18 24 25 30 31 35 36
19 25 26 31 32 36 37
20 26 27 32 33 37 38
21 27 28 33 34 38 39
22 28 29 34 35 39 40
23 29 30 35 36 40 41
24 30 31 36 37 41 42
25 31 32 37 38 42 43
26 32 33 38 39 43 44
27 33 34 39 40 44 45
28 34 35 40 41 45 46
29 35 36 41 42 46 47
30 36 37 42 43 47 48
31 37 38 43 44 48 49
32 38 39 44 45 49 50
33 39 40 45 46 50 51
34 40 41 46 47 51 52
35 41 42 47 48 52 53
36 42 43 48 49 53 54
37 43 44 49 50 54 55
38 44 45 50 51 55 56
39 45 46 51 52 56 57
40 46 47 52 53 57 58
41 47 48 53 54 58 59
42 48 49 54 55 59 60
43 49 50 55 56 60 61
44 50 51 56 57 61 62
45 51 52 57 58 62 63
46 52 53 58 59 63 64
47 53 54 59 60 64 65
48 54 55 60 61 65 66
49 55 56 61 62 66 67
50 56 57 62 63 67 68
51 57 58 63 64 68 69
52 58 59 64 65 69 70
53 59 60 65 66 70 71
54 60 61 66 67 71 72
55 61 62 67 68 72 73
56 62 63 68 69 73 74
57 63 64 69 70 74 75
58 64 65 70 71 75 76
59 65 66 71 72 76 77
60 66 67 72 73 77 78
61 67 68 73 74 78 79
62 68 69 74 75 79 80
63 69 70 75 76 80 81
64 70 71 76 77 81 82
65 71 72 77 78 82 83
66 72 73 78 79 83 84
67 73 74 79 80 84 85
68 74 75 80 81 85 86
69 75 76 81 82 86 87
70 76 77 82 83 87 88
71 77 78 83 84 88 89
72 78 79 84 85 89 90
73 79 80 85 86 90 91
74 80 81 86 87 91 92
75 81 82 87 88 92 93
76 82 83 88 89 93 94
77 83 84 89 90 94 95
78 84 85 90 91 95 96
79 85 86 91 92 96 97
80 86 87 92 93 97 98
81 87 88 93 94 98 99
82 88 89 94 95 99 100
83 89 90 95 96 100 101
84 90 91 96 97 101 102
85 91 92 97 98 102 103
86 92 93 98 99 103 104
87 93 94 99 100 104 105
88 94 95 100 101 105 106
89 95 96 101 102 106 107
90 96 97 102 103 107 108
91 97 98 103 104 108 109
92 98 99 104 105 109 110
93 99 100 105 106 110 111
94 100 101 106 107 111 112
95 101 102 107 108 112 113
96 102 103 108 109 113 114
97 103 104 109 110 114 115
98 104 105 110 111 115 116
99 105 106 111 112 116 117
100 106 107 112 113 117 118
101 107 108 113 114 118 119
102 108 109 114 115 119 120
103 109 110 115 116 120 121
104 110 111 116 117 121 122
105 111 112 117 118 122 123
106 112 113 118 119 123 124
107 113 114 119 120 124 125
108 114 115 120 121 125 126
109 115 116 121 122 126 127
110 116 117 122 123 127 128
111 117 118 123 124 128 129
112 118 119 124 125 129 130
113 119 120 125 126 130 131
114 120 121 126 127 131 132
115 121 122 127 128 132 133
116 122 123 128 129 133 134
117 123 124 129 130 134 135
118 124 125 130 131 135 136
119 125 126 131 132 136 137
120 126 127 132 133 137 138
121 127 128 133 134 138 139
122 128 129 134 135 139 140
123 129 130 135 136 140 141
124 130 131 136 137 141 142
125 131 132 137 138 142 143
126 132 133 138 139 143 144
127 133 134 139 140 144 145
128 134 135 140 141 145 146
129 135 136 141 142 146 147
130 136 137 142 143 147 148
131 137 138 143 144 148 149
132 138 139 144 145 149 150
133 139 140 145 146 150 151
134 140 141 146 147 151 152
135 141 142 147 148 152 153
136 142 143 148 149 153 154
137 143 144 149 150 154 155
138 144 145 150 151 155 156
139 145 146 151 152 156 157
140 146 147 152 153 157 158
141 147 148 153 154 158 159
142 148 149 154 155 159 160
143 149 150 155 156 160 161
144 150 151 156 157 161 162
145 151 152 157 158 162 163
146 152 153 158 159 163 164
147 153 154 159 160 164 165
148 154 155 160 161 165 166
149 155 156 161 162 166 167
150 156 157 162 163 167 168
151 157 158 163 164 168 169
152 158 159 164 165 169 170
153 159 160 165 166 170 171
154 160 161 166 167 171 172
155 161 162 167 168 172 173
156 162 163 168 169 173 174
157 163 164 169 170 174 175
158 164 165 170 171 175 176
159 165 166 171 172 176 177
160 166 167 172 173 177 178
161 167 168 173 174 178 179
162 168 169 174 175 179 180
163 169 170 175 176 180 181
164 170 171 176 177 181 182
165 171 172 177 178 182 183
166 172 173 178 179 183 184
167 173 174 179 180 184 185
168 174 175 180 181 185 186
169 175 176 181 182 186 187
170 176 177 182 183 187 188
171 177 178 183 184 188 189
172 178 179 184 185 189 190
173 179 180 185 186 190 191
174 180 181 186 187 191 192
175 181 182 187 188 192 193
176 182 183 188 189 193 194
177 183 184 189 190 194 195
178 184 185 190 191 195 196
179 185 186 191 192 196 197
180 186 187 192 193 197 198
181 187 188 193 194 198 199
182 188 189 194 195 199 200
183 189 190 195 196 200 201
184 190 191 196 197 201 202
185 191 192 197 198 202 203
186 192 193 198 199 203 204
187 193 194 199 200 204 205
188 194 195 200 201 205 206
189 195 196 201 202 206 207
190 196 197 202 203 207 208
191 197 198 203 204 208 209
192 198 199 204 205 209 210
193 199 200 205 206 210 211
194 200 201 206 207 211 212
195 201 202 207 208 212 213
196 202 203 208 209 213 214
197 203 204 209 210 214 215
198 204 205 210 211 215 216
199 205 206 211 212 216 217
200 206 207 212 213 217 218
201 207 208 213 214 218 219
202 208 209 214 215 219 220
203 209 210 215 216 220 221
204 210 211 216 217 221 222
205 211 212 217 218 222 223
206 212 213 218 219 223 224
207 213 214 219 220 224 225
208 214 215 220 221 225 226
209 215 216 221 222 226 227
210 216 217 222 223 227 228
211 217 218 223 224 228 229
212 218 219 224 225 229 230
213 219 220 225 226 230 231
214 220 221 226 227 231 232
215 221 222 227 228 232 233
216 222 223 228 229 233 234
217 223 224 229 230 234 235
218 224 225 230 231 235 236
219 225 226 231 232 236 237
220 226 227 232 233 237 238
221 227 228 233 234 238 239
222 228 229 234 235 239 240
223 229 230 235 236 240 241
224 230 231 236 237 241 242
225 231 232 237 238 242 243
226 232 233 238 239 243 244
227 233 234 239 240 244 245
228 234 235 240 241 245 246
229 235 236 241 242 246 247
230 236 237 242 243 247 248
231 237 238 243 244 248 249
232 238 239 244 245 249 250
233 239 240 245 246 250 251
234 240 241 246 247 251 252
235 241 242 247 248 252 253
236 242 243 248 249 253 254
237 243 244 249 250 254 255
238 244 245 250 251 255 256
239 245 246 251 252 256 257
240 246 247 252 253 257 258
241 247 248 253 254 258 259
242 248 249 254 255 259 260
243 249 250 255 256 260 261
244 250 251 256 257 261 262
245 251 252 257 258 262 263
246 252 253 258 259 263 264
247 253 254 259 260 264 265
248 254 255 260 261 265 266
249 255 256 261 262 266 267
250 256 257 262 263 267 268
251 257 258 263 264 268 269
252 258 259 264 265 269 270
253 259 260 265 266 270 271
254 260 261 266 267 271 272
255 261 262 267 268 272 273
256 262 263 268 269 273 274
257 263 264 269 270 274 275
258 264 265 270 271 275 276
259 265 266 271 272 276 277
260 266 267 272 273 277 278
261 267 268 273 274 278 279
262 268 269 274 275 279 280
263 269 270 275 276 280 281
264 270 271 276 277 281 282
265 271 272 277 278 282 283
266 272 273 278 279 283 284
267 273 274 279 280 284 285
268 274 275 280 281 285 286
269 275 276 281 282 286 287
270 276 277 282 283 287 288
271 277 278 283 284 288 289
272 278 279 284 285 289 290
273 279 280 285 286 290 291
274 280 281 286 287 291 292
275 281 282 287 288 292 293
276 282 283 288 289 293 294
277 283 284 289 290 294 295
278 284 285 290 291 295 296
279 285 286 291 292 296 297
280 286 287 292 293 297 298
281 287 288 293 294 298 299
282 288 289 294 295 299 300
283 289 290 295 296 300 301
284 290 291 296 297 301 302
285 291 292 297 298 302 303
286 292 293 298 299 303 304
287 293 294 299 300 304 305
288 294 295 300 301 305 306
289 295 296 301 302 306 307
290 296 297 302 303 307 308
291 297 298 303 304 308 309
292 298 299 304 305 309 310
293 299 300 305 306 310 311
294 300 301 306 307 311 312
295 301 302 307 308 312 313
296 302 303 308 309 313 314
297 303 304 309 310 314 315
298 304 305 310 311 315 316
299 305 306 311 312 316 317
300 306 307 312 313 317 318
301 307 308 313 314 318 319
302 308 309 314 315 319 320
303 309 310 315 316 320 321
304 310 311 316 317 321 322
305 311 312 317 318 322 323
306 312 313 318 319 323 324
307 313 314 319 320 324 325
308 314 315 320 321 325 326
309 315 316 321 322 326 327
310 316 317 322 323 327 328
311 317 318 323 324 328 329
312 318 319 324 325 329 330
313 319 320 325 326 330 331
314 320 321 326 327 331 332
315 321 322 327 328 332 333
316 322 323 328 329 333 334
317 323 324 329 330 334 335
318 324 325 330 331 335 336
319 325 326 331 332 336 337
320 326 327 332 333 337 338
321 327 328 333 334 338 339
322 328 329 334 335 339 340
323 329 330 335 336 340 341
324 330 331 336 337 341 342
325 331 332 337 338 342 343
326 332 333 338 339 343 344
327 333 334 339 340 344 345
328 334 335 340 341 345 346
329 335 336 341 342 346 347
330 336 337 342 343 347 348
331 337 338 343 344 348 349
332 338 339 344 345 349 350
333 339 340 345 346 350 351
334 340 341 346 347 351 352
335 341 342 347 348 352 353
336 342 343 348 349 353 354
337 343 344 349 350 354 355
338 344 345 350 351 355 356
339 345 346 351 352 356 357
340 346 347 352 353 357 358
341 347 348 353 354 358 359
342 348 349 354 355 359 360
343 349 350 355 356 360 361
344 350 351 356 357 361 362
345 351 352 357 358 362 363
346 352 353 358 359 363 364
347 353 354 359 360 364 365
348 354 355 360 361 365 366
349 355 356 361 362 366 367
350 356 357 362 363 367 368
351 357 358 363 364 368 369
352 358 359 364 365 369 370
353 359 360 365 366 370 371
354 360 361 366 367 371 372
355 361 362 367 368 372 373
356 3